On Efficient Zero-Knowledge Arguments

Zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften

von der KIT-Fakultät für Informatik
des Karlsruher Instituts für Technologie (KIT)

genehmigte
Dissertation

von
Michael Klooß
aus Karlsruhe

Tag der mündlichen Prüfung: 09. Januar 2023
Erster Gutachter: Prof. Dr. Jörn Müller-Quade
Zweiter Gutachter: Prof. Ph.D. Markulf Kohlweiss
Zusätzlicher Gutachter: Dr. Andy Rupp

Abstract

Proof systems and zero-knowledge arguments have been a captivating subject of research since their conception. Perhaps they receive even more attention nowadays, as they are reaching practical maturity. In this dissertation, we expand the understanding of zero-knowledge arguments, by providing new theoretical foundations, improved security analyses, and new constructions.

Theoretical Foundations. There are several definitions of zero-knowledge arguments, depending on the exact security guarantees. A widespread definition considers (strict) probabilistic polynomial time (PPT) provers, verifiers, and also adversaries, but allows *expected* polynomial time (EPT) simulator (and knowledge extractors). The asymmetry between the efficiency classes of simulator and adversary not only violates the idea of zero-knowledge — that anything that can be learned in an interaction with a prover could also be simulated without any interaction — but it also hinders sequential composability. Yet, this asymmetry is accepted in the plain model, because it is the only way to construct efficient zero-knowledge arguments of knowledge, i.e. constant round black-box protocols with negligible knowledge error and black-box simulation (Barak–Lindell, STOC'02). Compared to PPT, handling EPT adversaries is surprisingly non-trivial, since average-case efficiency is harder to handle and less well-behaved than worst-case efficiency. For example, an indistinguishable game hop can change the expectation from polynomial to infinite.

To rectify this behaviour, we define *computationally expected polynomial time (CEPT)* algorithms, which slightly expands the class of algorithms to include those which are indistinguishable from EPT algorithms. The resulting efficiency class not only behaves well w.r.t. indistinguishable game hops, it also allows us to handle CEPT adversaries with CEPT simulators, restoring the promise of zero-knowledge and enabling modular sequential composition.

Improved Security Analysis and Efficiency. Security of a cryptographic scheme, say a zero-knowledge argument Π, is usually defined asymptotically — in terms of negligible functions and polynomial time — and its security is shown by reductions to certain assumptions A_i. In this context, an important quantitative measure is the security loss, both in success probability, i.e. how much of the advantage of an adversary against Π can be turned into advantage against the assumptions A_i, and in runtime tightness, i.e. how does the runtime of the reduction to assumption A_i relate to the runtime of the attack on Π. Tightening the bounds on the security loss is both of theoretical and practical interest. Indeed, tight(er) reductions justify the use of small(er) security parameters and thus improve practical efficiency.

Related to such tightness questions is the "expressivity" of NP-relations. The more expressive a relation, the easier it is to encode high-level statements, and the smaller the witness (and often the security loss) can be. Working on the folding technique from Bulletproofs (Bootle et al., EUROCRYPT'16, and Bünz et al., S&P'18), we (1) define a new notion, called short-circuit extraction, to achieve tighter security for knowledge extraction; (2) derive an argument system whose witness relation is general quadratic equations (instead of the subclass of so-called rank-1 constraints) without compromising on efficiency.

Shorter Relaxed Range Proofs. Proving that a commitment contains a value which lies within a certain (integer) range is an important special-purpose relation, and such zero-knowledge arguments are called range proofs. For example, they are a widespread building block for privacy-enhancing technologies. We show how to translate the use of the four square decomposition (Boudot, EUROCRYPT'00) of non-negative integers, which was previously limited to integer commitment schemes over hidden order groups, to commitment schemes over groups of known prime order. Our most efficient schemes offer a relaxed form of soundness, namely membership within a range of rational numbers with short numerator and short denominator, which is sufficient for certain applications and more than 10× faster than Bulletproofs. We also show how to obtain stronger soundness, e.g. standard soundness, through addition of a single hidden order group element.

Zusammenfassung

Beweissysteme und zero-knowledge Argumente sind seit ihrer Erfindung ein faszinierender Forschungsgegenstand. Vielleicht erhalten sie heutzutage sogar mehr Aufmerksamkeit, da sie mittlerweile einen praktischen Reifegrad erreicht haben. In dieser Dissertation erweitern wir das Verständnis von zero-knowledge Argumenten, indem wir neue theoretische Grundlagen legen, verbesserte Sicherheitsanalysen geben und neue Konstruktionen aufzeigen.

Theoretische Grundlagen. Es gibt verschiedene Definitionen von zero-knowledge Argumenten, die sich in ihren genauen Sicherheitseigenschaften unterscheiden. Eine weit verbreitete Definition betrachtet strikte probabilistisch polynomialzeitige (PPT) Beweiser (prover), Prüfer (verifier), und ebensolche Angreifer, aber sie erlaubt *erwartet polynomialzeitige* (EPT) Simulatoren (und Wissensextraktoren (knowledge extractors)). Diese Asymmetrie zwischen der Effizienzklasse von Simulator und Angreifer verletzt nicht bloß die Idee der Zero-Knowledge-Eigenschaft — nämlich, dass ein Angreifer alles, was er in einer Interaktion mit dem Beweiser lernt, auch hätte ohne Interaktion selbst simulieren können — es behindert auch sequentielle Protokollkomposition. Nichtsdestotrotz ist diese Asymmetrie akzeptiert, da es im sogenannten schlichten Modell (plain model) die einzige Möglichkeit ist, effiziente zero-knowledge Wissensargumente (arguments of knowledge) mit Blackbox-Simulation zu konstruieren, d.h. Blackbox-Protokolle mit vernachlässigbarem Wissensfehler (knowledge error) und Blackbox-Simulation (Barak–Lindell, STOC'02). Im Vergleich zu PPT Angreifern ist es überraschend nicht-trivial EPT Angreifer zu behandeln, denn durchschnittliche (average-case) Laufzeit ist schwieriger handhabbar als schlimmstmögliche (worst-case) Laufzeit. Ein besonderes Problem ist hierbei, dass selbst ein ununterscheidbarer Spielschritt (game hop) die erwartete Laufzeit von polynomiell auf unendlich ändern kann.

Um dieses Verhalten zu korrigieren, definieren wir komplexitätstheoretisch (computationally) erwartete Polynomialzeit (CEPT), indem wir die Klasse der Algorithmen um solche erweitern, die Ununterscheidbar von EPT Algorithmen sind. Die daraus erhaltene Effizienzklasse verhält sich nicht nur gutartig bezüglich ununterscheidbarer Spielschritte, sie erlaubt es auch CEPT Angreifer mit CEPT Simulatoren zu handhaben, womit das Versprechen der Zero-Knowledge-Eigenschaft wiederhergestellt ist.

Verbesserte Sicherheitsanalysen und Effizienz. Die Sicherheit eines kryptographischen Verfahrens, z.B. eines zero-knowledge Arguments Π, ist üblicherweise asymptotisch definiert — über vernachlässigbare Funktionen und polynomielle Laufzeit — und dessen Sicherheit wird durch eine Reduktion auf bestimmte Annahmen A_i gezeigt. Ein wichtiges quantitatives Maß in diesem Kontext ist der Sicherheitsverlust, sowohl im Angreifererfolg — d.h. wie viel des Angreifervorteils in einen Vorteil gegen die Annahmen A_i umgewandelt wird — als auch in der relativen Laufzeit (runtime tightness) — d.h. wie sich die Laufzeit der Reduktion auf die Annahme A_i zur Laufzeit des Angriffs auf Π verhält. Schärfere Schranken für den Sicherheitsverlust sind sowohl von theoretischer als auch praktischer Bedeutung, denn schärfere Reduktionen ermöglichen die Verwendung niedrigerer Sicherheitsparameter womit die praktische Effizienz verbessert wird.

Verwandt mit diesen Fragen ist auch die „Ausdrucksstärke" einer NP-Relation. Je ausdrucksstärker eine Relation ist, desto einfacher ist es, eine abstrakte Aussage mithilfe der Relation zu codieren, und umso kleiner ist der Zeuge (und üblicherweise auch der Sicherheitsverlust). Unter diesem Gesichtspunkt betrachten wir die Faltungstechnik von Bulletproofs (Bootle et al., EUROCRYPT'16, and Bünz et al., S&P'18), und wir (1) definieren einen neuen Begriff, die Kurzschlussextraktion, die es erlaubt schärfere Laufzeitschranken für Wissensextraktion zu erreichen; (2) konstruieren ein Argumentsystem welches allgemeine quadratische Gleichungssysteme als Zeugenrelation besitzt (statt der Teilmenge R1CS wie bei Bulletproofs) ohne die Effizienz zu beeinträchtigen.

Kleinere Abgeschwächte Intervallbeweise. Sogenannte Intervallbeweise (range proofs) sind ein wichtiges Argumentsystem für eine bestimmte Relation. Sie beweisen, dass ein Wert in einem Commitment innerhalb eine bestimmten Ganzzahlintervalls liegt. Zum Beispiel in datenschutzfreundliche Technologien (privacy-enhancing technologies) sind Intervallbeweise ein weit verbreiteter Baustein. Wir zeigen auf, wie sich die Idee der Vier-Quadrate-Zerlegung (Boudot, EUROCRYPT'00) nicht-negativer Ganzzahlen, welche zuvor auf Ganzzahlcommitmentverfahren (integer commitment schemes) über Gruppen unbekannter Ordnung beschränkt waren, auf Commitmentverfahren über Gruppen bekannter primer Ordnung übertragen lassen. Unser effizientestes Verfahren bietet eine abgeschwächte Sicherheitsgarantie (relaxed soundness), denn statt Intervallzugehörigkeit als Ganzzahl wir nur Zugehörigkeit als Bruch mit kleinem Nenner und kleinem Zähler sichergestellt, was jedoch für bestimmte Anwendungen genügt und mehr als 10× schneller ist als Bulletproofs. Wir zeigen des Weiteren, wie stärkere Sicherheitsgarantien, z.B. gewöhnliche Sicherheit, mittels Hinzufügens eines einzigen Elements einer Gruppe unbekannter Ordnung erreicht werden kann.

Own Publications

[AFK22] Thomas Attema, Serge Fehr, and Michael Klooß. “Fiat-Shamir Transformation of Multi-Round Interactive Proofs”. In: *Theory of Cryptography - 20th International Conference, TCC 2022, Chicage, IL, USA, November 7-10, 2022, Proceedings, Part ?* Ed. by Eike Kiltz and Vinod Vaikuntanathan. Lecture Notes in Computer Science. To appear. Springer, 2022.

[CGKR22] Geoffroy Couteau, Dahmun Goudarzi, Michael Klooß, and Michael Reichle. “Sharp: Short Relaxed Range Proofs”. In: *CCS ’22: 2022 ACM SIGSAC Conference on Computer and Communications Security, Los Angeles, CA, USA, November 7 - 11, 2022.* Ed. by Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi. To appear. ACM, 2022. DOI: `10.1145/3548606.3560628`. URL: `https://doi.org/10.1145/3548606.3560628`.

[CKLR21] Geoffroy Couteau, Michael Klooß, Huang Lin, and Michael Reichle. “Efficient Range Proofs with Transparent Setup from Bounded Integer Commitments”. In: *Advances in Cryptology – EUROCRYPT 2021, Part III.* Ed. by Anne Canteaut and François-Xavier Standaert. Vol. 12698. Lecture Notes in Computer Science. Springer, Heidelberg, Oct. 2021, pp. 247–277. DOI: `10.1007/978-3-030-77883-5_9`.

[FBK+21] Sebastian H. Faller, Pascal Baumer, Michael Klooß, Alexander Koch, Astrid Ottenhues, and Markus Raiber. “Black-Box Accumulation Based on Lattices”. In: *18th IMA International Conference on Cryptography and Coding*. Ed. by Maura B. Paterson. Vol. 13129. Lecture Notes in Computer Science. Springer, Heidelberg, Dec. 2021, pp. 220–246. DOI: `10.1007/978-3-030-92641-0_11`.

[HHK+17] Gottfried Herold, Max Hoffmann, Michael Klooß, Carla Ràfols, and Andy Rupp. “New Techniques for Structural Batch Verification in Bilinear Groups with Applications to Groth-Sahai Proofs”. In: *ACM CCS 2017: 24th Conference on Computer and Communications Security*. Ed. by Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu. ACM Press, 2017, pp. 1547–1564. DOI: `10.1145/3133956.3134068`.

[HKR19] Max Hoffmann, Michael Klooß, and Andy Rupp. “Efficient Zero-Knowledge Arguments in the Discrete Log Setting, Revisited”. In: *ACM CCS 2019: 26th Conference on Computer and Communications Security*. Ed. by Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz. ACM Press, Nov. 2019, pp. 2093–2110. DOI: `10.1145/3319535.3354251`.

[HKRR20] Max Hoffmann, Michael Klooß, Markus Raiber, and Andy Rupp. “Black-Box Wallets: Fast Anonymous Two-Way Payments for Constrained Devices”. In: *Proc. Priv. Enhancing Technol.* 2020.1 (2020), pp. 165–194. DOI: `10.2478/popets-2020-0010`. URL: `https://doi.org/10.2478/popets-2020-0010`.

[Klo21] Michael Klooß. “On Expected Polynomial Runtime in Cryptography”. In: *TCC 2021: 19th Theory of Cryptography Conference, Part I.* Ed. by Kobbi Nissim and Brent Waters. Vol. 13042. Lecture Notes in Computer Science. Springer, Heidelberg, Nov. 2021, pp. 558–590. DOI: `10.1007/978-3-030-90459-3_19`.

[KLR19] Michael Klooß, Anja Lehmann, and Andy Rupp. "(R)CCA Secure Updatable Encryption with Integrity Protection". In: *Advances in Cryptology – EUROCRYPT 2019, Part I.* Ed. by Yuval Ishai and Vincent Rijmen. Vol. 11476. Lecture Notes in Computer Science. Springer, Heidelberg, May 2019, pp. 68–99. DOI: `10.1007/978-3-030-17653-2_3`.

Acknowledgements

First and foremost, I would like to thank my advisors Jörn Müller-Quade and Andy Rupp. I'm most grateful to Andy, for offering me the PhD position and for his constant encouragement; and I'm most grateful to Jörn, for giving me both freedom and support during my doctoral studies. All in all, I am very fortunate to having the both of you as my advisors. I am also sincerely grateful to Markulf Kohlweiss, for serving as second referee despite all the odds.

Secondly, I would like to thank all of my external co-authors: Carla Ràfols and Gottfried Herold for sharing their knowledge and teaching me that sometimes, less is more — an important lesson which I still struggle to apply; Max Hoffmann whose speed in implementing schemes, correcting them along the way, was never short of amazing (and I would be glad to be half as proficient); Anja Lehmann for her help and knowledge which steered a meandering project to completion with ease; Thomas Attema for not only answering open questions of mine, but also reaching out to me about them, leading to a fantastic paper; Serge Fehr for sharing his invaluable expertise, not only in cryptography, but also in handling some rather unusual circumstances during our joint project; Geoffroy Couteau, for his sharing boundless knowledge and dedication; Michael Reichle for his direct support and his creative solutions to any roadblock; Dahmun Goudarzi for his endurance in squeezing out optimal benchmark performance for our joint project; Huang Lin for providing and helping us theoreticians with real-world applications.

Moreover, I also would like to thank my hopefully soon-to-be co-authors: Carsten Baum for making sure my trip to Copenhagen was a complete success; Peter Scholl for hosting me at Aarhus and (taking over Carsten's role) patiently answering my questions; Chris Brzuska and Russell W.F. Lai for their warm welcome during my visit and for offering me the opportunity to continue my research as a post-doc with them.

Thirdly, I would like to thank all of my colleagues and all of the staff, former and present, old and new. In particular, I would like to thank: Akin Ünal for his dedication and unparalleled ability to make memorable moments; Alexander Koch for so many interesting conversations about research, philosophy and life, and his inspiring dedication to science and humanity in general; Astrid Ottenhues for generously lending me some of her incredible organizing abilities; Brandon Broadnax for proving that dedication to research is still possible outside of academia; Carmen Manietta for being the most reliable, efficient and kind secretary imaginable; Christian van Rensen for being an incredible help in writing lecture notes; Clemens Deußer for intriguing views and debates about politics and economy, and for some excellent food; Clemens Fruböse for many interesting conversations about (different approaches to) life; Dennis Hofheinz for being a helpful (runtime) oracle; Felix Dörre for all the amusing real-world (security) stories; Geoffroy Couteau for his inspiring dedication to research; Jeremias Mechler for sharing his incredible knowledge on composable security, his scrutiny which exposed so many problems before they propagated, and his invaluable help w.r.t. teaching; Lisa Kohl for her positivity and her advice; Marcel Tiepelt for his boundless optimism and motivation — without him my trip to Denmark might never have happened; Markus Raiber for sharing his knowledge about PETs in theory and practice, and for having perfected the art of endless (research) conversations with me; Matthias Nagel for showing me the ropes when I started and for a great vacation in the US; Nicholas Brandt for his help during the restructuring of the “topics in cryptography” lecture; Robin Berger for his invaluable help in and dedication to teaching and for providing me with enough time to complete my thesis and prepare my defense; Sebastian Faller for making me learn a lot as part of teaching, turning a student project into a publication along the way; Simon Hanisch, not only for setting up the IT infrastructure which got us through a pandemic, but also for organizing regular board game events; Sven Maier for making the institute more sportive and for many entertaining moments; Thomas Agrikola for organizing hiking trips and his help and advice in general; Valerie Fetzer for (co-)organizing recreational activities, ensuring a great vacation in Israel, her help with handicraft work, and taking care of so many little things which made my (and everyone else’s) life so much easier; Willi Geiselmann for supporting me in everything related to teaching.

I want to add a shout-out to my fellow stew-connoisseurs, everyone else joining the regular lunch, and the staff who makes the great stews.[1]

Finally, I would like to thank my family and friends. Thank you for enduring me — surely annoying at times — babbling about cryptography. Thank you for supporting me in focusing on my work, but even more so for distracting me from it. Especially to my parents, grandparents, and sister, I am most grateful that you had faith in me, maybe even more than I had myself. Knowing that I can always count on you made all of this so much more enjoyable.

Last but not least I wish to say,
to those who were left out,
if you believe I owe you thanks,
you can’t be wrong about [it].

To anyone who has been part
of this delightful quest,
I surely want to thank you here
and wish you all the best.

Now let me note a final thing:
I struggled — but conceded —
to make acknowledgements in rhyme,
I ended up defeated.

I truly tried and it was fun,
it went on for a while,
but as it happens all too oft,
I stopped for lack of time.

To me, just rhymes are hard enough,
as should be plain to see.
Observe that names are far more tough,
try Jörn Müller-Quade.

Away I threw my early draft,
went back where I set out.
If you succeed where I did not,
please let me know about [it].

Danksagung

Dieser Abschnitt gilt nun jenen,
die den oben nicht verstehen.
Von Herzen danke ich euch hier,
auch eingeschlossen manches Tier.
Denn euer Einsatz und Vertrau’n
war jener feste Unterbau,
der unverzichtbar — das ist klar —
für dieses Abenteuer war.

[1] Of course, this is not limited to the staff making the stews only. I’m grateful for (almost) all the food! And not just food, also my clean office, running water, and so much more.

Contents

Part I.

Introduction

1. Introduction

Since the invention of the personal computer, the internet and the world wide web, we have entered a new information age with easy, quick and omnipresent access to and distribution of digital information. However, these ever-present possibilities also gnaw away at the privacy of its voluntary and involuntary users. Cryptography provides the technical means to tackle the questions of privacy and authenticity in a digital world, enabling private and secure communication through encryption and digital signatures, and much more with other advanced cryptographic primitives and protocols, such as (fully) homomorphic encryption [RAD78; Gen09] or general multi-party computation[Yao86; GMW86].

Interactive Proofs and Arguments. Among these advanced cryptographic protocols, there are so-called *(interactive) argument systems*, also called *proof systems* [GMR85; BCC88],which generalize mathematical proofs. Traditionally, a mathematical proof of a statement $\mathbb{x}$ is a string $\mathbb{w}$ or sequence of logical derivations based on axioms (or previously established true statements). Thus, mathematical proofs can be interpreted as a relation $\mathcal{R}$ of statement $\mathbb{x}$ and proof $\mathbb{w}$, where $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$ if $\mathbb{w}$ is a convincing proof for $\mathbb{x}$. This idea is translated to complexity theory by taking into account the computational complexity of verifying a proof, hence considering an efficiently computable relation $\mathcal{R}$, i.e. an NP-relation, with associated language $\mathcal{L} = \{\mathbb{x} \mid \exists \mathbb{w} \colon (\mathbb{x}, \mathbb{w}) \in \mathcal{R}\}$ of "true statements" $\mathbb{x}$. The "proof" $\mathbb{w}$ is called a *witness* for $\mathbb{x} \in \mathcal{L}$ if $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$. In an interactive proof system, a party P, called the prover, wants to convince another party V, called the verifier, of the fact that $\mathbb{x} \in \mathcal{L}$.

A proof system should be *correct*, i.e. if two honest parties P and V interact on common input $\mathbb{x} \in \mathcal{L}$ then V should accept. It should be *sound*, i.e. if $\mathbb{x} \notin \mathcal{L}$ even a malicious prover P^* cannot convince the honest verifier V except with small probability. Proof systems for NP-languages with a probabilistic polynomial time (PPT) prover which receives a witness $\mathbb{w}$ as additional input (satisfying $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$), are also called *argument systems*. In cryptography, proof and argument system is often used interchangeably and merely refers to the syntax (i.e. a prover interacting with a verifier) of the protocol.[1]

While the trivial protocol, mimicking a mathematical proof by sending the witness $\mathbb{w}$ to the verifier, is a valid argument system, this does not exploit the power of interaction at all. Indeed, through interaction and computational complexity assumptions, completely new properties can be obtained:

Sublinear communication: The communication between prover and verifier can be much shorter than the witness $\mathbb{w}$, e.g. logarithmic in the bit length $|\mathbb{w}|$ of $\mathbb{w}$.

Sublinear computation: The computation of the verifier can be sublinear in $|\mathbb{w}|$ (but we do not consider this in this work).

[1] Historically, coming from complexity theory, proof systems were defined for arbitrary languages, not only NP-relations, and thus concerned unbounded provers (and consequently soundness against unbounded adversaries, i.e. *statistical* soundness). On the other hand, argument systems concerned (honest or malicious) PPT provers, and only guaranteed *computational* soundness. This distinction has mostly been replaced by explicit soundness guarantees (computational or statistical), especially in cryptography. Moreover, since even theoretical cryptography mostly revolves around efficient protocols, PPT provers for NP-relations are the archetypal setting.

Zero-knowledge: The interaction may only reveal that $\mathbb{x} \in \mathcal{L}$ but not leak any other information to a potentially malicious verifier V^*.

Knowledge soundness: A convincing prover must "know" a witness $\mathbb{w}$ with $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$ (even if $\mathbb{w}$ is not sent to the verifier, e.g. in a zero-knowledge argument).

An incredibly versatile combination of these properties are *zero-knowledge arguments of knowledge*, abbreviated as ZKAoK, which are argument systems that are both zero-knowledge and knowledge sound. For example, a ZKAoK allows the prover P to convince the verifier V that it honestly computed the output z of a public (efficient) program F with public input x and secret input w, i.e. that $z = F(x, w)$. Almost anything of interest (indeed, anything one can compute) is easily expressed as a public program with public and secret inputs, e.g. one can prove that a ciphertext contains a solution to a hard puzzle, say a Sudoku, without revealing the solution. Knowledge soundness asserts that P really knows such a witness w. Zero-knowledge asserts that, besides what can be derived from the output $z = F(x, w)$, nothing (more) about w is revealed. This privacy-preserving assurance of honest computation is what makes ZKAoKs an extremely flexible and broadly applicable tool. In practice, one uses relations and protocols which are tailored to specific problems of interest. For example, an identification card might prove that one is older than 18 years without revealing the actual date of birth.

Before we jump to the contributions of this thesis, let us recall on a high-level how zero-knowledge arguments of knowledge are defined, and how they are proven secure.

1.1. Zero-Knowledge Arguments (of Knowledge)

The requirements of zero-knowledge and (knowledge) soundness seem contradictory. But these two guarantees consider different possible misbehaviours, and thus can co-exist. For zero-knowledge, one must control what a potentially malicious verifier learns from the interaction with an honest prover. For knowledge soundness, one must extract a witness from a potentially malicious prover which convinces an honest verifier. Now, we give a more precise, but still informal, discussion of these properties.

Zero-Knowledge Simulator. What does it mean to "learn nothing" from an interaction? The general definition of (black-box) zero-knowledge [GMW86] postulates an efficient *simulator* Sim which is given black-box rewinding access (denoted by $\mathsf{Sim}^{\mathsf{V}^*}$) to the potentially malicious verifier V^*, i.e. Sim can send messages to a virtual copy of V^*, receive responses and rewind it to some prior state. The goal is that $\mathsf{Sim}^{\mathsf{V}^*}(\mathbb{x})$ generates an output which is indistinguishable from the view of V^* (i.e. all messages V^* received) in an interaction with the honest prover $\mathsf{P}(\mathbb{x}, \mathbb{w})$. Observe that Sim neither has $\mathbb{w}$ as input nor interacts with $\mathsf{P}(\mathbb{x}, \mathbb{w})$, yet its simulated output is indistinguishable from the view of V^*. In other words, instead of interacting with P, V^* could just run $\mathsf{Sim}^{\mathsf{V}^*}$ in its head to obtain the same information. Thus, clearly, nothing new is learnt by interacting with the prover P.

To achieve this feat, the simulator Sim uses its black-box rewinding access to V^* to explore V^*'s reaction to different messages in a given state, by sending a message, rewinding V^*, and then trying another message. This power of the simulator over the adversary allows manipulating an execution (via rewinding) in a way which is completely impossible in a normal, straight-line execution. For example, the simulator might first learn a verifier's next message, called a *challenge*, rewind it, and then produce a specially crafted response which is convincing for exactly that challenge (but maybe none other).

Knowledge Extractor. The definition of (black-box) knowledge soundness [BG93; BG11] postulates an efficient *extractor* Ext which is given black-box rewinding access to some potentially malicious prover P^*. Exploiting the power of rewinding, the extractor is able to obtain *multiple related responses* from P^*, by feeding it different challenges (in place of the honest verifier). In a knowledge sound protocol, these responses are correlated in such a way that it is possible to recover the witness.

Rewinding: Two Sides of the Same Coin. In our description of zero-knowledge simulation resp. knowledge extraction, we considered black-box rewinding access to the adversary (the malicious verifier, resp. the malicious prover). While, this is conceptually simple and clean, it also comes with a price: It can be shown that *practically efficient* protocols, namely *constant round* protocols with negligible soundness error, are *impossible* with PPT simulation and extraction. To avoid this, a widespread definition of zero-knowledge arguments of knowledge allow *expected polynomial time* (EPT) simulators and/or extractors. This leads to neat definitions, allows proving natural protocols secure, and seems like a wonderful solution. Alas, unlike PPT, EPT does not compose well and is prone to subtle problems.

Setups for Practical Efficiency. In the above, we always considered the *plain model*, where prover and verifier interact without any common/shared information except the statement $\mathbb{x}$. For practically efficient zero-knowledge arguments, one often relies on a so-called *setup assumption*, e.g. that prover and verifier share a *common reference string* (CRS) which was generated by an honest party. In the best case, this CRS is simply a *uniformly random string* (URS), which can be heuristically generated in the real world by nothing-up-my-sleeve methods, e.g. using the digits of π. These setups also allow simulation in PPT. In principle, they also allow extraction in PPT, however, to the best of our knowledge this is only possible when the proofs do not have sublinear communication (excluding a large class of proofs of particular interest) or under strong assumptions.[2]

Tightness of Security. One can define zero-knowledge simulation and knowledge extraction using quantitative measures. For zero-knowledge, one can quantify simulation in terms of a simulation error (for the simulation's quality) and the runtime of the simulator. For knowledge soundness, one can quantify in terms of a knowledge error (which ties V's acceptance and Ext's extraction probability together) and the runtime of the extractor. In theoretical works, the runtime tightness, i.e. the factor by which the runtime of a simulator or extractor increased over the real execution, is often ignored (beyond ensuring polynomial time). However, to justify a *concrete choice* of security parameterλ, one cannot argue with "negligible" and "PPT" anymore, since these make no sense for fixed λ. Thus, to have good provable assurances in this setting, it is vital to achieve the fastest possible runtime for simulator and extractor, as well as the best possible quality guarantees in terms of simulation and knowledge error, respectively. In practice, using setup assumptions facilitates very simple and efficient simulation. Indeed, it is typically as fast as the adversary (hence has constant runtime tightness) and

[2] Common idealized models are the random oracle model (ROM) [BR93] and the generic group model (GGM) [Sho97; Mau05], and there are argument systems which are zero-knowledge and straight-line knowledge sound in these models. More generally, knowledge assumptions are used which lie in between full idealization (like ROM and GGM) and standard assumptions. On the one hand, such knowledge assumptions facilitate efficient extraction and often yield the tightest plausible proofs in practice. On the other hand, they are also a very strong class of assumption and many of them contradict the existence of indistinguishability obfuscation (IO) [BCPR16], which has since been established from well-founded assumptions [JLS21]. Yet, some knowledge assumptions can, in some sense, be realized by IO [AHK20]. The idealized models do not suffer from such contradictions but are known to be uninstantiable in general [CGH98; GK03].

the simulated view is statistically indistinguishable from the real view. However, the same does not hold for extraction, especially not for sublinear communication argument systems.[3]

1.2. Handling Expected Polynomial Time

As noted above, for efficient zero-knowledge proofs (in the plain model), expected polynomial time (EPT) simulation and extraction is a necessity [BL04]. However, the standard definition only considers *PPT* adversaries. This introduces an asymmetry in the complexity classes of zero-knowledge simulator and adversary, violating the intuitive promise that everything an adversary learns in a zero-knowledge protocol (in the plain model) it could just simulate itself. Even worse, it makes it impossible to exploit simple (sequential) composition properties, because once a PPT adversary V^* is replaced by a simulation $\mathsf{Sim}^{\mathsf{V}^*}$, the new adversary $\mathsf{V}' = \mathsf{Sim}^{\mathsf{V}^*}$ is now EPT and thus cannot be handled by a zero-knowledge simulator or knowledge extractor anymore. While it is possible to prove sequential self-composition directly, it is not so simple for more general forms of composition, especially in the context of multi-party computation where (sequential) composition of protocols is particularly useful.

Lastly, when trying to define what expected time adversaries are, one has many choices. For example: Does the time of the *honest* prover count? Does the time required to compute the auxiliary adversarial inputs matter (or do we assume non-uniform auxiliary inputs, which do not consume computation time)? Does the to-be-called-efficient adversary V^* have to be "efficient" (say EPT) only when it is run with the honest prover P, or must it be "efficient" when run with *any* "efficient" (say EPT) party P^*? The last question is the most crucial one, and we call the former type of adversaries *designated* adversaries, since they only need to be efficient in the protocols they are designed to attack, but not in other, arbitrary contexts. This is the most natural definition, but also the hardest to handle.

Related work. Already Feige [Fei90] provides a discussion and example in his thesis which demonstrates the difficulties in defining and handling designated adversaries. In particular, Feige shows that the naive idea of simply cutting off the adversary's execution after an a priori fixed number of steps fails for any a priori polynomial cutoff (under plausible assumptions). Despite pointing out these problems, Feige decides to keep the standard definition. Katz and Lindell [KL05; KL08] are the first to provide a solution, which is however not fully general. Their solution assumes the security of primitives against superpolynomial-time adversaries, hence requires superpolynomial hardness assumptions. The use of superpolynomial hardness allows *superpolynomial* cutoffs, sidestepping Feige's obstructions. Subsequently, [Gol07; Gol10] presented another solution, which intuitively works by postulating "nice" adversaries. He considers a complexity class which (by definition) contains only algorithms whose runtime cannot become superpolynomial even when exploiting black-box rewinding access to "attack" it.[4] At the expense of shrinking the class of adversaries against which security is proven, this makes the difficulties of rewinding expected time and designated adversaries disappear, by essentially excluding potentially problematic algorithms by definition. While Hofheinz, Unruh, and Müller-Quade [HUM13] deal with strict polynomial time, i.e. PPT, they need to handle *designated* PPT adversaries (something which is usually avoided or unnecessary) and develop techniques on which we build in our work.

[3] Knowledge assumptions and idealized models (cf. Footnote 2) circumvent this, but we do not consider these.

[4] Consider the interactive algorithm A which draws a random string $r \xleftarrow{\$} \{0,1\}^\lambda$, then expects a message s, then sends r back, and if $r = s$ it runs an extra 2^λ steps. In an interactive context, the expected time of A is $\mathcal{O}(\lambda)$, since, unless r was guessed and sent to A, it just sends r and terminates immediately. With rewinding, one can first send $s \coloneqq 0^\lambda$ and thus learn r from A, then rewind A and then deliberately send $s = r$, making A run for 2^λ steps *always*. Goldreich's definition forbids such algorithms.

Contribution. In Chapter 5, we define a relaxation of expected polynomial time, called computationally expected polynomial time (CEPT). Intuitively, we consider a runtime distribution T CEPT if it is computationally indistinguishable from a runtime distribution T' which is EPT in the usual sense. We prove a convenient characterization of CEPT which shows a, perhaps surprising, equivalence between statistical and computational indistinguishability for runtimes. Moreover, we show that with CEPT, we can finally handle designated CEPT adversary with CEPT simulation. This leads not only to symmetry for adversary and simulator in the definition of zero-knowledge arguments which includes the efficient black-box constant round arguments we aimed for. But it also shows that the most natural definition of efficient adversary, a *designated* adversary, can indeed be handled. Contrary to Goldreich [Gol10], instead of shrinking the class of allowed adversaries, we ever so slightly broaden it (and with it also the class of allowed simulators), and unlike Katz and Lindell [KL08] we do not require superpolynomial hardness assumptions to prove security.

1.3. Revisiting Efficient Log-Size Arguments from Hardness of DLOG

For practically efficient zero-knowledge arguments, one usually accepts a setup assumption in the form of a CRS. In theory, the CRS is simply generated by a trusted party which disposes of any potential trapdoor information that was needed during CRS generation (say, the factorization (p, q) if the CRS is an RSA modulus $n = pq$). In practice, generating a CRS poses a difficult problem because the incentive to keep and exploit the above-mentioned trapdoor information is huge. Thus, it is best to have a *transparent* setup, that is, a CRS which is simply a uniformly random string (URS). Such a URS can be heuristically obtained from nothing-up-my-sleeve methods, e.g. using digits of π, hashing a newspaper, using sunspots [CPs07], and so on.

One of the most prominent examples of a practically efficient proof system with very small logarithmic-size communication and transparent setup are so-called Bulletproofs [BBB+18], an optimization of [BCC+16]. As this part of the work is focused on understanding and improving Bulletproofs, we give a high level overview of their characteristics.

Bulletproofs use a CRS as setup, which consists entirely of random group elements in a prime order group with hard DLOG assumption, in particular, the setup is transparent. The CRS is used to instantiate a linear commitment scheme. A commitment scheme allows to commit to a value v such that v is fixed but hidden (similar to encryption), until it is (potentially) unveiled. A commitment scheme where it is possible to add commitments (implicitly adding the committed values) and multiply commitments with a scalar (implicitly multiplying the committed values with a scalar) is called *linear*. Linear commitment schemes are one of the main tools for practically efficient (zero-knowledge) proofs of knowledge.

The crucial building block of Bulletproofs is a logarithmic communication inner product argument (IPA), i.e. an argument (which is *not* zero-knowledge) for proving that $\langle \boldsymbol{x}, \boldsymbol{y} \rangle = t$ for committed $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{F}_p^n$, $t \in \mathbb{F}_p$, where the statement $\mathbb{x} = (c, t)$ consists of a commitment c to $(\boldsymbol{x}, \boldsymbol{y})$ and of t, and the witness is $\mathbb{w} = (\boldsymbol{x}, \boldsymbol{y})$. On a very high level, logarithmic communication (in $|\mathbb{w}|$) is achieved by repeatedly halving and linearly recombining committed vectors. Since commitments are constant size, each halving has constant communication.[5] When n hits 1, the witness $(\boldsymbol{x}, \boldsymbol{y})$ is sent in the plain. This technique is called *folding*.

The NP-relation for which [BBB+18] is designed are so called *rank 1 constraint systems* (R1CS) [BCG+13]. Given a vector $\boldsymbol{w} \in \mathbb{F}_p^n$, the witness, a R1CS is a system of equations of the form $(\sum_{i=1}^n a_i w_i) \cdot (\sum_{i=1}^n b_i w_i) =$

[5] Strictly speaking, constant in the dimension of the committed vector (but not in the security parameter).

$\sum_{i=1}^{n} c_i w_i$. Given linear commitments and an IPA (without zero-knowledge) as above, [BBB+18] reduces verification of a committed R1CS solution to verification of an inner product, sending only a constant number of commitments.

Contribution. In Chapter 4, we revisit the Bulletproofs construction, trying to understand, generalize and optimize it further. For our approach, we need a *zero-knowledge* IPA and we show how to construct it from the Bulletproofs IPA with very little overhead. Moreover, we found a simple optimization which halves the prover's computation and enables other standard optimizations. However, our main contributions are the following:

- Relying on R1CS as a relation means that, perhaps surprisingly, Bulletproofs cannot prove inner product statements without overhead. We rectify this by constructing $\mathrm{QESA}_{\mathrm{ZK}}$, an argument system which proves *general quadratic equation systems* (QE) which contain R1CS as a special case. In general, comparing different argument systems for different languages is a complicated question, cf. Appendix B.8. But fortunately, $\mathrm{QESA}_{\mathrm{ZK}}$ is both faster and smaller than [BBB+18], even if [BBB+18] uses our improved IPA. So, there is not much to debate about the improved expressivity gained from handling QE instead of R1CS in this specific comparison.
- We outline a path towards a tighter knowledge soundness reduction. For this, we devised a generalized form of the so-called *special soundness* property, which allows what we call *short-circuit extraction.* With this, we show that a much smaller number of related transcripts suffice for extraction than previously known. Concretely, our tightest instantiation needs a linear number of transcripts (whereas Bulletproofs use a quadratic number [BBB+18]). We complement this result by a lower-bound for certain hard languages, which even if it may not apply to QEs, shows that to avoid an almost linear number of related transcripts, novel proof techniques will be necessary.

Lastly, our new proof system is flexible and can be combined easily with other proof systems in the same setting (i.e. linear commitments based on DLOG). We demonstrate this swapping out the proof system in the proof of correctness of a shuffle of ElGamal ciphertexts by Bayer and Groth [BG12], yielding the first practically efficient protocol with logarithmic communication. This example relies on another building block, a so-called linear map preimage argument, which is obtained by the folding approach and implicit in the IPA.

Related and Subsequent Work. We restrict our attention to the most closely related works, and refer to Chapter 4 for more discussions. Clearly, our work is based on Bulletproofs [BCC+16; BBB+18]. A close subsequent work is [CHJ+22], whose approach also relies on a *zero-knowledge* inner product argument (zk-IPA), but they only handle R1CS.

The work [JT20] provides a tighter analysis of the extractor in [BCC+16], improving the knowledge soundness. In [ACK21] a modified extractor is shown to be essentially optimal for special sound protocols (both in knowledge soundness and runtime). Neither of the works [JT20; ACK21] exploit short-circuit extraction, resulting in quartic runtime tightness for Bulletproofs and $\mathrm{QESA}_{\mathrm{ZK}}$ (in the witness size) for general constraints, whereas short-circuit extraction suggest that linear runtime tightness is possible for $\mathrm{QESA}_{\mathrm{ZK}}$. Another line of works [GT21; GOP+22] analyzes the security of Bulletproofs (with Fiat–Shamir transformation applied) using the algebraic group model [FKL18], which is a (strong) knowledge assumption. Perhaps (not) surprisingly, for algebraic adversaries, they achieve essentially optimal tightness results (both in knowledge error and runtime) since their extraction is straight-line, i.e. does not rewind the adversary.

1.4. Short(er) Relaxed Range Proofs in the DLOG Setting

In Chapter 3 we present a novel range proof[6] in the DLOG setting. A range proof is a special-purpose zero-knowledge argument of knowledge that shows that a committed value lies in some range $[A, B] \subseteq \mathbb{Z}$. This is a crucial building block for many applications, especially in privacy-preserving protocols such as anonymous credentials [Cha90; CL01].

In the DLOG setting, one works modulo the group order n, i.e. over $\mathbb{Z}_n$. Although it makes sense to define an "integer range $[A, B]_{\mathbb{Z}_n}$" in $\mathbb{Z}_n$ as the image of the set $[A, B]$ modulo n, it is preferable to choose a system of representatives and only consider ranges within that. The usual choice of representatives is $\{-\lfloor \frac{n-1}{2} \rfloor, \dots, \lfloor \frac{n-1}{2} \rfloor\} \subseteq \mathbb{Z}$, as it also gives a sensible notion of positive, negative, and shortness in $\mathbb{Z}_n$.

For efficient range proofs based on linear commitments, there are two general approaches:

k-ary decomposition: The idea is to reduce the general statement "$v \in [0, B]$" to a simpler statement by k-ary decomposition. Usually, binary decomposition is used, i.e. one proves $v = \sum_{i=0}^{\lceil \log(B+1) \rceil} 2^i b_i$ and $b_i \in \{0, 1\}$. If n is prime then for $b \in \mathbb{Z}_n$ we get $b \in \{0, 1\} \iff b(1-b) \equiv_n 0$. As a low-degree polynomial equation, this is efficiently provable in zero-knowledge via linear commitments.

Square decomposition: Over the *integers*, for any $x \in \mathbb{Z}$, the square x^2 is non-negative. Moreover, every non-negative number v can be decomposed efficiently as the sum of four squares [RS86; PS19]. Thus, proving that $v \cdot (B - v) \geq 0$ by decomposition into 4 squares ensures $v \in [0, B]$ over the integers.

The k-ary decomposition approach is very easy to use, integrates easily into many protocols, and is adequately efficient. Indeed, Bulletproof range proofs [BBB+18] use the binary decomposition approach to prove range membership. The biggest downside of binary decomposition is that many values (namely all b_i) must be committed to, making the protocols relatively computationally expensive.

The deceptively simple square decomposition approach, however, makes no sense modulo n and requires to work *over the integers*. Thus, linear *integer commitments* are required. Such commitments exist, but (group-based) constructions rely on factoring-based assumptions, and are relatively large and relatively computationally expensive as a consequence.

Contribution. In Chapter 3, we present a range proof over $\mathbb{Z}_p$, for prime p. It is inspired by so-called relaxed/approximate proofs of short preimages from lattice-based cryptography, and offers a slightly weakened soundness guarantee: Instead of considering (short) *integer* representatives of $\mathbb{Z}_p$ as used in the explanation above, we consider *short rational representatives*, that is, fractions $\frac{n}{d}$ with short numerator n and short denominator d. This allows us to obtain a protocol whose communication (i.e. proof size) is very small and which is highly efficient. Indeed, computation-wise it easily outperforms Bulletproofs,[7] with the prover being more than 10× faster.

Shortness of rational representatives modulo p is in general incompatible with shortness of integer representatives, e.g. $\frac{p+1}{2} \in \mathbb{Z}_p$ has a large integer representative, but the short rational representative $\frac{1}{2} \in \mathbb{Z}_p$. Moreover, shortness of rational representatives does not behave well under addition, since denominators can grow multiplicatively. Thus, one might question the usefulness of this relaxed

[6] As noted in Footnote 1, "proof" and "argument" is used interchangeably in cryptography and in this work. The term "range argument" is much less common, even if it is technically correct (and perhaps more precise).

[7] For proving range membership of a single commitment, even the proof size is comparably small. For larger batch range proofs, the proof size grows linearly, unlike the logarithmic growth for Bulletproofs.

soundness. However, if it is known *a priori* that a representative is a short integer, then proving membership in the interval of short rational representatives actually proves membership in the interval of short *integer* representatives. In other words, given this prior knowledge, relaxed soundness is upgraded to standard soundness. A prime application example is a digital identification card, where the birth date is known to lie within a predetermined range (say the last 200 years).

Nevertheless, we also provide a simple and efficient way to augment our range proof with a single RSA group element to obtain standard soundness, slightly increasing proof size and computation, and unfortunately, losing the transparent setup. Alternatively, using so-called class groups (of hidden order) is compatible with transparent setup and forces denominators of the form 2^k (for small k), making rational representatives behave well under addition, eliminating an important source of trouble.

Chapter 3 is based on [CGKR22a] which is a follow-up of our work [CKLR21b] and aimed specifically at the DLOG setting, improving usability by using the relaxed soundness notion, improving efficiency by changing the basic protocol for proving the decomposition, modularizing the decomposition and shortness proofs, and improving applicability by using a novel batch proof of shortness (derived from a "core lemma") to efficiently handle standard sized groups,[8] e.g. 256-bit order elliptic curves.

Related Work. Many range proofs are based on the k-ary decomposition, which is certainly the most straightforward approach in prime order groups. Concrete examples are the works [CCs08; Gro11] and of course, the Bulletproofs family of range proofs [BBB+18; CHJ+22]. Of course, due to the simplicity of the approach, many works use binary decomposition within their protocols.

Range proofs based on square decomposition of integers are for example [Bou00; Lip03; Gro05; CPP17]. As noted, these (must) rely on integer commitment schemes [FO97; DF02] based on a group $\mathbb{H}$ of hidden order. Prior work computes (almost) the full proofs over $\mathbb{H}$. These works are well-suited for very large ranges. For small ranges, say 64-bit integers, which are of great practical importance, working (solely) in $\mathbb{H}$ leads to comparatively large proof sizes, since hidden order group elements must be large to withstand generic attacks for computing (a multiple of) the group order. Conversely, in elliptic curves, λ-bit security against DLOG requires only 2λ-bit elliptic curves.

Our range proof is inspired by lattice-based constructions of so-called relaxed/approximate proofs of short preimages (e.g. [Lyu09; Lyu12]), so there are clear similarities. However, to the best of our knowledge, the proof of our core lemma, and indeed the specific setting and usage, does not appear in the lattice-based literature and (standard) lattice-based techniques do not seem immediately applicable.

1.5. Other Published Work

This section briefly summarizes other work published during my doctoral studies, but which is not included in this thesis. The work [HHK+17], which evolved from my master's thesis, is omitted.

[8] The work [CKLR21b] needs either group sizes which depend on the range, or it needs rather expensive (parallel) repetitions to boost the soundness error.

1.5.1. Black-box Accumulators: Privacy-Preserving Incentive Systems

Part of my work was on privacy-preserving "point collection" systems, dubbed black-box accumulation [JR16; HHNR17], which very roughly, allow a user to receive and spend points. This abstracts a form of electronic payments with clearly separated roles of system operator and user, e.g. to implement customer loyalty programs, reward programs, or anonymous ticket systems. The system operator, as the name suggests, operates the system and wants to ensure that the points it gives out to users, or points which users spend, behave as expected. That is, a user can collect a number of points, and it can spend the collected points, but it can never spend more points than it collected. The user wishes to use the system privately, hence the receive and spend operations should not reveal the user's identity.

Contribution. In the work [HKRR20], we improve upon the efficiency of [HHNR17] by avoiding bilinear groups, using blind signatures, a security proof in the generic group model (GGM), and (linear-size) Bulletproofs for security. The (linear-size) Bulletproofs can in fact be replaced by our new relaxed range proofs [CGKR22a], leading to another significant speed-up. In another work [FBK+21], we translated the construction of [HHNR17] to the lattice setting, which required lattice-specific adaptations to the general template to remain reasonably efficient without counting the overhead of the zero-knowledge arguments.

Related Work. Closely related works in general are anonymous credentials (AC) [Cha90; CL01], which are an anonymous form of identification scheme which allows to prove additional properties in zero-knowledge, e.g. that one is over 18 years old and a college student. There are many flavours and efficient instantiations [BL13; RVH17] of ACs, e.g. keyed-verification ACs [CMZ14; CR19], updatable ACs [BBDE19; BEK+20], and, of course, black-box accumulation schemes [JR16; HHNR17]. So-called "central bank digital currencies" (CBDC) are another relatively new and closely related building block, though setting, scope of application, and requirements [BEB+] (e.g. anti-money laundering mechanisms) are different. Cryptographic techniques used for privacy-preserving CBDCs [WKDC22; KKS22] are closely related to (keyed verification, updatable) ACs and BBA. Electronic cash, introduced by Chaum [Cha82], is yet another closely related primitive, in a different setting. Instead of system operator and user, there are now banks, merchants and users. Users buy from merchants by "spending" coins, and merchants later "deposit" the spent coins in the bank to receive the money. Thus, despite sharing basic techniques, constructions [CFT98; CHL05; BCFK15] and challenges differ from AC/BBA/CBDC.

1.5.2. (R)CCA Secure Updatable Encryption with Integrity Protection

In many contexts, it is mandated to secure encrypted data by periodic "key rotation", i.e. an update of the encryption keys for the encrypted data. To study this question theoretically, the notion of updatable encryption (UE) schemes has been defined. Very roughly, an updatable encryption scheme comes with two additional algorithms: A key-update algorithm, which takes as input an old key k and outputs a new key k' and an update token Δ, and a ciphertext update algorithm, which takes as input an old ciphertext c under key k and an update token Δ, and outputs a new ciphertext c' (which is encrypted under k'). The idea is that the server, which stores all ciphertexts, can update them after receiving the update token Δ without further interaction. In fact, there are two flavours of updatable encryption: The one described above has ciphertext-*independent* update tokens, and another flavour has ciphertext-*dependent* update tokens, where the update token may depend on a (small) ciphertext header.

As security notions, one uses variants of privacy resp. integrity notions, adapted to the setting and called UE-IND-CPA/CCA, i.e. indistinguishability under chosen plain/ciphertext attacks, resp. UE-INT-PTXT/CTXT, i.e. plain/ciphertext integrity. In the adapted security games, the adversary may (repeatedly) cause updates of keys and ciphertexts and it may corrupt keys and update tokens, unless this causes it to trivially win the game. Depending on the definition of trivial wins, ciphertext updates obtain a *directionality*; indeed, in many constructions it is possible to use an update token to both upgrade and downgrade ciphertexts. The UE-specific security notion UE-IND-UPD, indistinguishability of updates, ensures that ciphertexts (of the same size) become indistinguishable after a ciphertext update. For technical reasons, UE-IND-CCA (and UE-INT-CTXT) are only defined for *deterministic* updates, and so-called replayable CCA security (RCCA) [CKN03] is the strongest known option for probabilistic updates.

Contribution. In our work [KLR19], we define the notions of (bidirectional) updatable security with integrity protection for ciphertext-*independent* schemes and present two constructions. The first construction efficiently achieves deterministic UE-IND-CCA, UE-INT-CTXT and UE-IND-UPD. The second construction achieves (probabilistic) UE-IND-RCCA, UE-INT-PTXT and UE-IND-UPD, but is of mostly theoretical interest due to heavy reliance on Groth–Sahai proofs [GS08] and their malleability [CKLM12].

Related Work. Prior work by Everspaugh, Paterson, Ristenpart, and Scott [EPRS17] considered the ciphertext-*dependent* setting and achieved indistinguishability and integrity notions for this case. The work by Lehmann and Tackmann [LT18] introduced the ciphertext-*independent* setting, but only defined and achieved UE-IND-CPA and UE-IND-UPD. Subsequent works revisited and extended our security definitions [Jia20; Nis21; BMPR21], and several works study the deterministic setting with UE-IND-CCA security, providing more efficient constructions [BDGJ20; BEKS20]. Constructions for unidirectional schemes have also been found [SS21; MPW22; GP22]. Moreover, RCCA-secure updatable encryption has been used to construct secure onion routing with replies [KHRS21]. To our knowledge, there was no progress on more efficient probabilistic updatable encryption with integrity protection, leaving our quite inefficient construction as the only RCCA-secure option.

1.5.3. Fiat–Shamir Transformation of Multi-Round Special Sound Protocols

The Fiat–Shamir transformation converts interactive arguments of knowledge into non-interactive arguments by letting the prover compute the verifier's challenges as the hash of the partial transcript. It is well-known that the transformation is provably secure in the random oracle model when applied to certain 3-move protocols, or more generally constant round protocols. Pathological examples are known which show that, in general, the security loss is exponential in the number $2\ell + 1$ of rounds, assuming the prover moves first, i.e. assuming ℓ challenges.

Contribution. We show in [AFK22] that for multi-round *special sound* protocols the knowledge error and extraction runtime tightness of the Fiat–Shamir transformed proof is only $(Q + 1)$-fold of that of the interactive proof, where Q bounds the number of queries to the random oracle. In particular, the security loss is independent of the number $2\ell + 1$ of rounds. Multi-round special sound protocols are a broad class which, for example, includes Bulletproofs. It is easy to see that our result is essentially best possible (due to concrete attacks). Moreover, we demonstrate a *non-pathological* attack on the Fiat–Shamir transformation when applied to a t-fold parallel repetition of certain special sound protocols.

Assuming for simplicity that ℓ divides t and Q, the knowledge error grows at least by a factor of $Q^\ell/\ell^{t+\ell}$ (compared to the t-fold parallel repetition of the interactive protocol), hence exhibits an exponential loss in the number $2\ell + 1$ of rounds.

Related Work. In concurrent work by Wikström [Wik21], a similar positive result was found, but presented and derived with a distinctly different terminology and point of view. Ben-Sasson, Chiesa, and Spooner [BCS16] define *state-restoration soundness* (SRS) and *knowledge* (SRK), an abstract notion for interactive arguments which essentially corresponds to their Fiat–Shamir transformation, and the respective SRS and SRK error is equivalent to that of the Fiat–Shamir transformed proof. The notions of *round-by-round soundness* [CCH+19] and *round-by-round knowledge* [CMS19] have also been introduced to study Fiat–Shamir transformations, but are essentially straight-line notions and do not seem applicable to multi-round special sound protocols in general.

1.6. Structure of the thesis

The thesis is divided into three parts. The first part begins with this introduction in Chapter 1. It continues with Chapter 2, which contains the basic notation and common definitions that are used throughout this thesis. The second part contains the main content.

- In Chapter 3, we present our new relaxed range proof construction in the DLOG setting.
- In Chapter 4, we present our results concerning optimizations of the folding technique, the zero-knowledge inner product argument, our quadratic satisfiability argument. Moreover, it contains our notion of short-circuit extraction.
- In Chapter 5, we introduce computationally expected polynomial time (CEPT) and show how it allows us to handle designated adversaries and achieve symmetry between runtime classes of adversary and simulator.

Each of these chapters begins with a brief outline of the contributions of the respective authors (if applicable). We conclude this part with an outlook and open questions in Chapter 6.

The third part is the appendix, where further discussions, detailed proofs, and definitions of lesser importance can be found. The Appendices A, B, and C correspond to Chapters 3, 4, and 5, respectively.

2. Preliminaries

This chapter provides unified preliminaries for the rest of this work. Definitions, remarks, examples and discussions are taken (sometimes verbatim) from the papers [HKR19a; Klo21; CGKR22a] or their respective full versions [HKR19b; Klo20; CGKR22b].

2.1. Notation and Basic Functions

We use log for the binary logarithm. We write $[a, b]$ for an interval $[a, b]$ in $\mathbb{Z}$, and we write $[a, b]_R$ for an interval in another space R, e.g. $\mathbb{Q}, \mathbb{R}, \mathbb{Z}_p$. We denote by $|x|$ the absolute value of $x \in \mathbb{R}$. For a string $s \in \{0, 1\}^*$, we denote by $|s|$ its bitlength.

For a randomized algorithm $\mathcal{A}$ with input x, we write $y \leftarrow \mathcal{A}(x; r)$ for its execution with explicit randomness r. If the randomness is not explicit, we write $y \leftarrow \mathcal{A}(x)$ and assume that r was sampled accordingly. We write $s \overset{\$}{\leftarrow} S$ for sampling s uniformly at random from a finite set S or $d \leftarrow D$ to sample d randomly according to a given probability distribution D. For simplicity, we assume that it is possible to draw uniformly at random from any set $\{1, \dots, n\}$ for $n \in \mathbb{N}$.[1] Further, we generally assume that some public parameters, denoted by pp, and the security parameter, denoted by λ, are implicitly passed as input to algorithms if it is clear by the context. By poly we denote some (arbitrary but fixed) polynomial, and by negl we denote some negligible function, i.e. a function with $\lim_{\lambda \to \infty} \lambda^c \mathsf{negl}(\lambda) = 0$ for any $c \in \mathbb{N}$.

For interactive algorithms $\mathsf{A}_1, \mathsf{A}_2$, we write $\langle \mathsf{A}_1(x, z), \mathsf{A}_2(y, z) \rangle$ for the execution where A_1 and A_2 interact, given private input x resp. y and common input z. The sequence tr of exchanged messages during the interaction is called the **transcript** and denoted by $tr \leftarrow \langle \mathsf{A}_1(x, z), \mathsf{A}_2(y, z) \rangle$. We write $\mathsf{out}_{\mathsf{A}_i} \langle \mathsf{A}_1, \mathsf{A}_2 \rangle$ for the **output** of A_i, where we have omitted the inputs to A_1 and A_2 for simplicity. Similarly, the **view** $view_i$ of A_i is denoted by $\mathsf{view}_{\mathsf{A}_i} \langle \mathsf{A}_1, \mathsf{A}_2 \rangle$ and defined as a tuple which contains all information available to A_i, i.e. its inputs (x, z) resp. (y, z), its random tape r_i, and all messages m_ℓ it received during the execution. Hence, given the view, one can replay the execution of A_i. Black-box rewinding access to an algorithm $\mathsf{A}(x, z)$ (with fixed inputs x, z) works by first sampling and fixing the random tape r, and then providing the next-message function as an oracle, i.e. the function which takes as input a sequence of messages $(m_1, \dots, m_\ell)$ and returns the response $\mathsf{A}(x, z)$ would send if it received these messages (in that order) from its communication partner.[2] For giving algorithm B black-box rewinding access to A with (fixed) input x, we write $\mathsf{B}^{\mathsf{A}(x,z)}$, or $\mathsf{B}^{\mathsf{bbrw}(\mathsf{A}(x,z))}$ if we need to be very explicit.

[1] This subtlety is often ignored in the literature. With any a priori bounded finite number of steps and only binary random coins it is impossible to sample uniformly from $\{1, \dots, n\}$ unless n is a power of 2. However, it can be approximated exponentially precisely, see also Section 2.3.2.1.

[2] If a message sequence makes no sense, e.g. because A halts after processing a single message but $\ell \geq 2$, then the next message function returns a special symbol $\perp$ indicates an error state for this input. More specific discussion on machine models, interaction models, their subtle effects and the robustness of most notions and definitions under sensible concrete choices of these models, can be found in Chapter 5.

For Chapters 3 and 4 we use following simple notion of probabilistic polynomial time (PPT) and expected polynomial time (EPT) algorithms: A non-interactive algorithm A is (classically) PPT (resp. EPT) if it is PPT (resp. EPT) in its total input length $|x|$, i.e. there exists a polynomial poly such that the total (expected) number of steps of the computation $\mathsf{A}(x)$ is bounded by $\mathsf{poly}(|x|)$ for any $x \in \{0,1\}^*$. An *interactive* algorithm A is PPT (resp. EPT) if it is PPT (resp. EPT), if its total (expected) number of steps is polynomially bounded in its *first input*, which is the implicit input 1^λ unless specified otherwise. In other words, for interactive algorithms, we assume an a priori polynomial bound $\mathsf{poly}(\lambda)$ on the number of steps. This definition is good enough for our purposes, even though it is not general enough to cover all reactive systems of interest, see e.g. the discussion on runtime definitions in [HUM13].

Remark 2.1.1 (Non-uniformity). The usual definition of algorithm, i.e. a finite program specification, is also called *uniform.* The *non-uniform* setting considers infinite specifications, which in our case means the circuit family P/poly or equivalently PPT machines with an additional (infinite) advice *advc*.

In the rest of this work, non-adversarial (protocol) parties are *always* uniform. For adversarial parties, all of our definition make sense for both uniform and non-uniform adversaries (perhaps after a minor modification). Since we only use the adversary in a black-box manner (and all reductions in this thesis are uniform), the presence or absence of advice is irrelevant. Indeed, we usually do not pass an explicit advice *advc* as input to the adversary. Recall that universal quantification over inputs (e.g. in IND-CPA), often implies non-uniform hardness. For uniform hardness, possible inputs must also be generated (efficiently). This mostly affects Chapter 5, where we deal with notions of efficiency for expected time adversaries. There, the power of non-uniformity trivializes some reductions, but we provide uniform reductions to ensure our efficiency notions do not depend on non-uniformity.

2.1.1. Probability Theory

By U_X we denote the uniform distribution on a finite set X.

Definition 2.1.2. Let μ, ν be two probability measures on a *countable* set S. We define the **statistical distance** as

$$\Delta(\mu,\nu) = \sup_{A\subseteq S} \mu(A) - \nu(A) = \frac{1}{2}\sum_{a\in A} |\mu(\{a\}) - \nu(\{a\})|.$$

We define the **sup-ratio** $\rho_{\mathsf{sup}}(\mu/\nu)$ as

$$\rho_{\mathsf{sup}}(\mu/\nu) = \sup_{A\subseteq S} \mu(A)/\nu(A) = \sup_{s\in \mathrm{supp}(\mu)} \mu(s)/\nu(s)$$

where $0/0 = 1$ and $x/0 = \infty$ for $x > 0$.

We recall some important properties of the sup-ratio: Given two random variables X and Y and any set of outcomes S, we have

$$\Pr[X \in S] \leq \rho_{\mathsf{sup}}(X/Y) \cdot \Pr[Y \in S].$$

Consequently, for real-valued X, Y, resp. for an arbitrary (measurable) function f, we find

$$\mathbb{E}[X] \leq \rho_{\mathsf{sup}}(X/Y) \cdot \mathbb{E}[Y] \qquad \text{resp.} \qquad \rho_{\mathsf{sup}}(f(X), f(Y)) \leq \rho_{\mathsf{sup}}(X, Y).$$

We will make ample use of these two facts. Moreover, we use that

$$\rho_{\mathsf{sup}}((X', Y')/(X, Y)) \leq \rho_{\mathsf{sup}}(X'/X) \cdot \rho_{\mathsf{sup}}(Y'/Y)$$

for pairs (X', Y') (resp. (X, Y)) of *independent* random variables.

Remark 2.1.3. Let $0 < d \leq C$ be integers and let $\gamma \xleftarrow{\$} [0, C-1]$ and $u' \xleftarrow{\$} [0, d-1]$. Suppose $u = \gamma \bmod d$. Then it is easily seen that $\rho_{\mathsf{sup}}(u/u') \leq 1 + d/C$. This follows, e.g., from $\rho_{\mathsf{sup}}(\gamma/\gamma') \leq 1 + d/C$ where $\gamma' \leftarrow [0, d\lceil C/d \rceil]$ and noting that $u' = \gamma' \mod d$ in distribution.

2.2. Cryptographic Primitives

We define syntax and semantics of cryptographic primitives, and sketch their security properties.

2.2.1. Cryptographic Groups

We work in the DLOG setting with cryptographic groups. In general, we write $\mathbb{G}$, $\mathbb{H}$, etc., for groups and use capital letters G, H, etc., for group elements. All cryptographic groups are cyclic (hence commutative) and we use *additive* notation, i.e. we write $G + H$ and $x \cdot G$ or xG for $G, H \in \mathbb{G}$, $x \in \mathbb{Z}$. We denote by $\langle G \rangle$ the cyclic subgroup generated by G. Despite additive notation, we sometimes speak of *exponents* and *exponentiations* instead of *scalars* and *scalar-group-multiplications*, especially for efficiency considerations.

Remark 2.2.1 (Implicit representation in prime order groups). In Chapter 4, we only consider prime-order groups and use a special notation, namely implicit representation of group elements by their dlog w.r.t. an (implicit) distinguished generator in $\mathbb{G}$. That is, we write $[1] \in \mathbb{G}$ for a fixed generator and $[z] := z \cdot [1]$ for $z \in \mathbb{Z}_p$ for arbitrary group elements. We stress that given $[z]$, the DLOG z may be hard to compute, i.e. the representation $[z]$ is only a notational simplification and z is only *implicitly* specified, hence the name **implicit representation**.

A PPT algorithm GrpGen on input 1^λ outputs a (description of a) group $\mathbb{G} = \mathbb{G}_\lambda$. Given the description, group operations (addition and inverse) and membership tests are efficient, as well as bounds $U_{\mathsf{lo}} \leq |\mathbb{G}| \leq U_{\mathsf{up}}$ on the group order are specified. By $A \xleftarrow{\$} \mathbb{G}$ we denote a uniformly random group element. When we say "$\mathbb{G}$ is a group of (prime) order $p = p_\lambda$", we mean that $p = |\mathbb{G}|$ is known unless explicitly stated otherwise.

Remark 2.2.2 (Random generators). For simplicity, we use random generators in hardness assumptions. While using deterministic generators is possible, it would affect certain reductions and require (small) adaptions and appropriately strengthened assumptions throughout.

In general, we define the hard DLOG assumption for groups as follows.

Definition 2.2.3 (DLOG assumption). The **DLOG assumption** in a group $\mathbb{G}$, or more precisely, for group generator GrpGen, holds if for every PPT adversary $\mathcal{A}$, the advantage

$$\mathsf{Adv}^{\mathsf{dlog}}_{\mathcal{A}}(1^\lambda) := \Pr[\mathbb{G} \xleftarrow{\$} \mathsf{GrpGen}(1^\lambda); G \xleftarrow{\$} \mathbb{G} \setminus \{0\}, H \xleftarrow{\$} \mathbb{G}; x \leftarrow \mathcal{A}(1^\lambda, \mathbb{G}, G, H) \colon H \stackrel{?}{=} x \cdot G]$$

is negligible.

Notation 2.2.4. For notational simplicity, we leave GrpGen implicit in the rest of the work. Moreover, we usually omit indexing group $\mathbb{G}$ and group order p by the security parameter, as already practised in Definition 2.2.3. Typically group descriptions are part of the public parameters *pp*.

The experiment in Definition 2.2.3 is only efficient if it is possible to efficiently sample uniformly random elements G, H from the cryptographic group $\mathbb{G}$. More generally, one can consider any sampling algorithm Sample for elements in $\mathbb{G}$ and define hardness of DLOG w.r.t. Sample. Moreover, one can distinguish two notions, where one notion keeps the random coins r of Sample secret or and the other makes them public, i.e. passes r as input to the adversary. In most groups of interest, we can assume that it is possible to sample uniformly at random with public coins, essentially by interpreting a random bitstring as a group element. In particular, DLOG hardness for public and secret random coins is equivalent in that case. More generally, this is true if Sample is reverse sampleable. We refer to the discussion in Section 2.3.2 for further details. Non-trivial sampling will only be of interest for assumptions in class groups, namely in Appendix A.1.

Example 2.2.5. In groups of prime order p, it is easy to sample uniformly from $\mathbb{G}$ given a generator G, i.e. an element $G \in \mathbb{G} \setminus \{0\}$. Simply pick $x \overset{\$}{\leftarrow} \mathbb{Z}_p$ and set $H = x \cdot G$. In implicit notation, we would have $G = [1]$ and $H = x \cdot [1] = [x]$. Hardness of DLOG now translates to hardness of finding x given $([1], [x])$.

The concrete assumptions used in Chapter 3 and in Chapter 4 are generalizations of the DLOG assumption in different directions. We recall them in the respective preliminaries.

2.2.2. Hash Functions

Definition 2.2.6 (CRHF). Let $\mathsf{Hash}\colon \mathcal{K}_\lambda \times \{0,1\}^* \mapsto \{0,1\}^{\ell(\lambda)}$ be a hash function. We call Hash a **collision-resistant** hash function (CRHF), if for all PPT adversaries $\mathcal{A}$ there exists a negligible function negl such that

$$\Pr\left[\begin{matrix} k \overset{\$}{\leftarrow} \mathcal{K}_\lambda; (m_0, m_1) \leftarrow \mathcal{A}(1^\lambda, k)\colon \\ m_0 \neq m_1 \wedge \mathsf{Hash}(k, m_0) = \mathsf{Hash}(k, m_1) \end{matrix}\right] \leq \mathsf{negl}(\lambda).$$

Recall that, due to generic birthday attacks, we need at least $\ell = 2\lambda$ output size for λ bits of security. Moreover, *keyed* hash functions are required to achieve collision-resistance against *non-uniform* adversaries, otherwise advice could contain collisions. The keys are part of the public parameters and omitted until noted otherwise.

2.2.3. Random Oracle Model

In the random oracle model (ROM), introduced by Bellare and Rogaway [BR93], all parties have oracle access to a truly random function $\mathsf{RO}\colon \{0,1\}^* \to \{0,1\}^{2\lambda}$. In this model, it is possible to provide (much) more efficient constructions for many primitives, e.g. highly efficient signatures via the Fiat–Shamir transformation [FS87] from passively secure identification schemes or via hash-then-sign constructions from trapdoor permutations and preimage samplable functions [BR96; GPV08] as well as highly efficient CCA secure encryption from IND-CPA secure encryption schemes via the Fujisaki–Okamoto transformation [FO99]. Heuristic instantiations of constructions which were proven secure in the ROM have fared well in practice [KM15], though it is well-known that this heuristic is unsound and insecure in general [CGH98; GK03]. In fact, even constructions for primitives which are provably impossible in the CRS model are possible in the ROM, such as non-interactive non-committing encryption [Nie02].

Fiat–Shamir Transformation. The Fiat–Shamir transformation (for proof systems) converts public-coin proofs (of knowledge) to non-interactive zero-knowledge (proofs of knowledge) (NIZK resp. NIZKPoK) by computing the verifier's challenges as random oracle hashes over partial transcripts and other context information (which includes $\mathbb{x}$). In case of non-zero correctness error, one retries in case of aborts [Lyu09]. In practice, the ROM is heuristically instantiated by a strong cryptographic hash function, e.g. SHA-3. Note that a URS can be generated trivially in the ROM. For multi-round special sound protocols (Section 2.5), the work [AFK22] intuitively shows, that the knowledge error (resp. runtime) of an extractor is increased by a factor of Q compared to the knowledge error (resp. runtime) of the extractor for the interactive protocol.

2.2.4. Non-Interactive Commitments

A **(non-interactive) commitment scheme** COM allows committing to a message $m \in \mathcal{M}$ obtaining a commitment $c \in \mathcal{C}$ and opening information $r \in \mathcal{R}$, where $\mathcal{M}$, $\mathcal{C}$, $\mathcal{R}$ are message, commitment and opening (or randomness) space of COM, respectively. A commitment scheme should be *hiding* and *binding*. The hiding property ensures that a commitment reveals nothing about the committed value. The binding property ensures that the committed value cannot be changed, i.e. that there is at most one value to which a commitment can be unveiled. We now define the security properties formally.

Definition 2.2.7. A **(non-interactive) commitment scheme** COM consists of PPT algorithms (Setup, Com, VfyOpen), which behave as follows:

- $\mathsf{Setup}(1^\lambda, pp) \to ck$: The PPT algorithm Setup takes as input the security parameter λ (in unary) and public parameters pp, and outputs a commitment key ck.
- $\mathsf{Com}(1^\lambda, pp, ck, m; r) \to c$: The PPT algorithm Setup takes implicit inputs 1^λ and pp, a commitment key ck and a message m (in the messages space $\mathcal{M}_{ck}$) and outputs a commitment c to m. The randomness r is also called the *opening*.[3]
- $\mathsf{VfyOpen}(1^\lambda, pp, ck, c, r, m) \to b$: The PPT algorithm VfyOpen takes implicit input 1^λ and pp, a commitment key ck, a commitment c with opening r and a messages m. It outputs a bit b, indicating whether the tuple (c, r, m) is a valid commitment or not.

In the rest of this work, we will usually omit the standard inputs $(1^\lambda, pp)$ to all of these algorithms. We often write $\mathsf{Com}_{ck}(m; r)$ and $\mathsf{VfyOpen}_{ck}(c, m, r)$ as the commitment key is usually fixed over many commitments.

Definition 2.2.8 (Correctness). A commitment scheme COM = (Setup, Com, VfyOpen) is **(perfectly) correct**, if for any $ck \xleftarrow{\$} \mathsf{Setup}(1^\lambda)$, any message $m \in \mathcal{M}_{ck}$ and any $c \leftarrow \mathsf{Com}_{ck}(m; r)$, it holds that $\mathsf{VfyOpen}_{ck}(c, r, m) = 1$.

Definition 2.2.9 (Hiding Property). The advantage of a stateful adversary $\mathcal{A}$ against the hiding property of a commitment scheme COM is

$$\mathsf{Adv}^{\mathrm{hide}}_{\mathcal{A}}(\lambda) = \Pr\left[\begin{array}{c} pp \leftarrow \mathsf{GenPP}(1^\lambda);\ ck \leftarrow \mathsf{Setup}(1^\lambda, pp);\ b \xleftarrow{\$} \{0,1\}; \\ (m_0, m_1) \leftarrow \mathcal{A}(1^\lambda, pp, ck);\ c \leftarrow \mathsf{Com}_{ck}(m_b; r); \\ b' \leftarrow \mathcal{A}(c) \colon b' = b \end{array}\right]$$

[3] The opening is often defined to be an output of Com. We will not need this more general definition.

A commitment scheme COM is **(computationally) hiding** if for every stateful PPT adversary $\mathcal{A}$, there exists a negligible function negl such that $\mathsf{Adv}^{\mathrm{hide}}_{\mathcal{A}}(\lambda) \leq \frac{1}{2} + \mathsf{negl}(\lambda)$. It is **statistically hiding** if the same holds for unbounded adversaries.

Remark 2.2.10 (Multi-Hiding Property). The hiding experiment can be changed to allow the adversary to obtain many challenge commitments, by allowing it to repeatedly query the experiment for a pair (m_0, m_1) and receiving $\mathsf{Com}_{ck}(m_b)$ (for the same choice of b in all queries). By a standard hybrid argument, an adversary $\mathcal{A}$ with advantage ε against the multi-hiding property which makes at most Q challenge queries induces an adversary $\mathcal{B}$ against the standard hiding property with advantage at least ε/Q and runtime distribution essentially identical to that of $\mathcal{A}$ (except for the small bookkeeping overhead associated with hybrid distinguishers).

Definition 2.2.11 (Binding Property). The advantage of a stateful adversary $\mathcal{A}$ against the binding property of a commitment scheme COM is

$$\mathsf{Adv}^{\mathrm{hide}}_{\mathcal{A}}(\lambda) = \Pr\left[\begin{array}{c} pp \leftarrow \mathsf{GenPP}(1^\lambda);\ ck \leftarrow \mathsf{Setup}(1^\lambda, pp); \\ (c, r_0, r_1, m_0, m_1) \leftarrow \mathcal{A}(1^\lambda, pp, ck): \\ m_0 \neq m_1 \land \mathsf{VfyOpen}_{ck}(c, r_0, m_0) = 1 \\ \land\ \mathsf{VfyOpen}_{ck}(c, r_1, m_1) = 1 \end{array}\right]$$

A commitment scheme COM is **(computationally) binding** if for every stateful PPT adversary $\mathcal{A}$, there exists a negligible function negl such that $\mathsf{Adv}^{\mathrm{bind}}_{\mathcal{A}}(\lambda) \leq \mathsf{negl}(\lambda)$. It is **statistically binding** if the same holds for unbounded adversaries.

2.3. (Non-)Interactive Proof Systems

In this section, we define proof systems and their properties. As noted before (Chapter 1 Footnote 1), we use proofs system and argument system interchangeably. Thus, in this section, we opt for proof systems instead of argument systems for the naming.

2.3.1. Proof Systems and Soundness

Before defining proof systems, we recall (parameter-dependent) relations and languages. A *param*-**dependent relation** $\mathcal{R}$ over $\mathcal{X} \times \mathcal{Y}$ is a family of relations $\mathcal{R}_{param} \subset \mathcal{X} \times \mathcal{Y}$. We say $\mathcal{R}$ is a **PPT relation** or **NP relation** if there is a PPT algorithm which given *param* decides if a pair $(\mathbb{x}; \mathbb{w})$ is in $\mathcal{R}_{param}$. The *param*-**dependent language** of $\mathcal{R}$ is defined as the family $\mathcal{L}_{param} = \{\mathbb{x} \in \mathcal{X} \mid \exists \mathbb{w} : (x; w) \in \mathcal{R}_{param} = 1\}$. Instead of families of relations (resp. languages), we can consider tuples $(param, \mathbb{x}, \mathbb{w})$ as elements in $\mathcal{R}$, where $(param, \mathbb{x}, \mathbb{w}) \in \mathcal{R} \iff (\mathbb{x}, \mathbb{w}) \in \mathcal{R}_{param}$.

Convention 2.3.1. Usually, relations are *pp*-dependent (or *crs*-dependent). As usual, we often omit the dependency in our notation to keep the visual noise low. Adding the dependencies on *pp* is a straightforward mechanical process.

Now, we define proof systems.[4] We always consider a common reference string (CRS) as setup, even though some proof systems do not require them. Moreover, we only define *efficient* proof systems for PPT relations (i.e. NP relations).

[4] In Chapter 5, we give another definition of proof systems and associated notions. But that is to generalize to efficiency notions beyond PPT. Using it here would needlessly complicate Chapters 3 and 4.

Definition 2.3.2 (Proof system for $\mathcal{R}$). Let $\mathcal{R}$ be a PPT relation over a set $\mathcal{X}$, defining a language $\mathcal{L}$. A **proof system** is a tuple $(\mathsf{GenCRS}, \mathsf{P}, \mathsf{V})$ of algorithms, where GenCRS is a PPT algorithm and (P, V) is a pair of interactive PPT algorithms. They behave as follows:

- $\mathsf{GenCRS}(1^\lambda, pp) \to crs$: The PPT algorithm GenCRS takes as input the security parameter λ (in unary) and public parameters, and outputs a common reference string *crs*.
- $\mathsf{P}(1^\lambda, pp, crs, \mathbb{x}, \mathbb{w})$: This interactive PPT algorithm takes as input a CRS *crs* and a statement-witness pair $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$, and interacts with an instance of $\mathsf{V}(crs, \mathbb{x})$. There is no output.
- $\mathsf{V}(1^\lambda, pp, crs, \mathbb{x}) \to b$: This interactive PPT algorithms takes as input a CRS *crs* and a statement $\mathbb{x}$ (not necessarily in $\mathcal{L}$), and output a verdict $b \in \{0, 1\}$. If $b = 1$ we say V *accepts* the statement $\mathbb{x}$, else it *rejects*.

Proof systems for parameter-dependent relations and languages (e.g. *pp*- or *crs*-dependent) are defined in the obvious way.

In the rest of this work, we will usually omit the input 1^λ (and often *pp*) to all of these algorithms.

Notation 2.3.3. We write $tr \leftarrow \langle \mathsf{P}(s), \mathsf{V}(t) \rangle$ for the *transcript* of an interaction where P (resp. V) has input s (resp. t) and *implicit inputs* $1^\lambda, pp, crs$. We write $b = \langle \mathsf{P}(s), \mathsf{V}(t) \rangle$ for the verifier's output b.

To handle techniques such as rejection sampling, we allow a non-negligible correctness error in our proof systems.

Definition 2.3.4 (Correctness). A proof system $(\mathsf{GenCRS}, \mathsf{P}, \mathsf{V})$ for $\mathcal{L}$ has **correctness error** γ_{cor} if for every adversary $\mathcal{A}$

$$\Pr\begin{bmatrix} pp \leftarrow \mathsf{GenPP}(1^\lambda); crs \leftarrow \mathsf{GenCRS}(pp); \\ (x, w) \leftarrow \mathcal{A}(pp, crs): \\ \langle \mathsf{P}(pp, crs, x, w), \mathsf{V}(pp, crs, x) \rangle = 1 \end{bmatrix} \geq 1 - \gamma_{\mathrm{cor}}(\lambda)$$

We call $(\mathsf{GenCRS}, \mathsf{P}, \mathsf{V})$ **correct** if $\gamma_{\mathrm{cor}} = \mathsf{negl}$. It is **perfectly** correct if $\gamma_{\mathrm{cor}} = 0$.

Unless stated otherwise, we assume that any proof system we consider has negligible correctness error.

Definition 2.3.5 (Soundness Error). Let $\Pi = (\mathsf{GenCRS}, \mathsf{P}, \mathsf{V})$ be a public coin proof system for NP-relation $\mathcal{R}$ with language $\mathcal{L}$. Let $\mathcal{A}$ be a probabilistic algorithm and P^* be a deterministic algorithm. Then

$$\mathsf{Adv}^{\mathsf{snd}}_{\mathsf{P}^*,\mathsf{V}} = \Pr\begin{bmatrix} pp \leftarrow \mathsf{GenPP}(1^\lambda); crs \leftarrow \mathsf{GenCRS}(pp); \\ (x, s) \leftarrow \mathcal{A}(pp, crs); \\ b = \langle \mathsf{P}^*(x, s), \mathsf{V}(x) \rangle : \ b = 1 \wedge x \notin \mathcal{L} \end{bmatrix}$$

is the advantage of $(\mathcal{A}, \mathsf{P}^*)$ in the soundness experiment. We say Π has **statistical soundness error** δ_{snd}, if $\mathsf{Adv}^{\mathsf{snd}}_{\mathsf{P}^*,\mathsf{V}}(\lambda) \leq \delta_{\mathrm{snd}}(\lambda)$ for any (unbounded) pair of algorithms $(\mathcal{A}, \mathsf{P}^*)$, called the adversary. We say Π has **computational soundness error** δ_{snd}, if for any PPT pair of algorithms $(\mathcal{A}, \mathsf{P}^*)$ there exists a negligible function negl such that $\mathsf{Adv}^{\mathsf{snd}}_{\mathsf{P}^*,\mathsf{V}}(\lambda) \leq \delta_{\mathrm{snd}}(\lambda) + \mathsf{negl}$.

2.3.2. Reverse Sampling, Public-Coin, and Transparent Setup

We first give a general definition of efficient sampling and reverse sampling, and then use this to define a generalized notion of public-coin proof systems and transparent setup.

2.3.2.1. Efficient and Reverse Sampling

The central requirement for making interactive proofs non-interactive by computing the verifier's messages "locally" via some means (e.g., a random oracle) is that V should not keep any secrets during the protocol, as the non-interactive prover interacts with a "virtual" verifier in its head, and necessarily knows its state. In particular, the verification checks (w.l.o.g. all happening in the very last verifier step) cannot depend on any secrets, i.e., checks can only depend on the (virtual) transcript of the interaction. The usual definition of public-coin interactive proofs ensures these requirements by having the verifier pass out parts of its random tape. However, this simplifies the situation beyond what is actually practical: Verifier random coins are binary (hence they don't draw from arbitrary challenge sets $\mathcal{C}_i$, such as $\mathcal{C}_i = \mathbb{Z}_p^\times$) and likewise random oracles, when instantiated with hash functions, output bitstrings. Here, we discuss how to bridge this gap. For this, we use the security parameter denoted λ (which may be set to $\lambda = |x|$), and a parameter η which controls the quality of sampling algorithms (which may be set to λ).

Definition 2.3.6. Let $\eta \in \mathbb{N}$ and $\mathcal{X} \subset \{0,1\}^*$. A family of distributions $(D_x)_{x\in\mathcal{X}}$ is **efficiently sampleable** (from the uniform distribution $(U_{\mathsf{poly}(\eta,|x|)})_{\eta\in\mathbb{N},x\in\mathcal{X}}$), if there is a PPT algorithm M such that $X_{\eta,x} = M(1^\eta, x; U_{\mathsf{poly}(\eta,|x|)})$ is distributed as D_x when conditioned on $X_{\eta,x} \neq \bot$, and $\Pr[X_{\eta,x} = \bot] \leq 2^{-\eta}$.

Let M efficiently sample $(D_x)_{x\in\mathcal{X}}$. Then M is **efficiently reverse sampleable**, if there is a PPT algorithm $M^\dagger$ such that $M^\dagger$ efficiently samples $(R_{\eta,(x,y)})_{\eta\in\mathbb{N},x,y\in\{0,1\}^*}$, where $R_{\eta,(x,y)}$ is uniform over $\{r \in \{0,1\}^{\mathsf{poly}(\eta,|x|)} \mid M(1^\eta, x; r) = y\}$. That is, $M^\dagger(1^\eta, (x,y))$ samples uniformly from random tapes r with $y = M(1^\eta, x; r)$ (except with probability $2^{-\eta}$).

Reverse sampling is sometimes called **invertible sampling**.

Efficiently sampleable distributions can be approximated exponentially precise and sampling is errorless in the sense that an output $\bot$ indicates failure, and retries to amplify success are possible. The failure probability is to account for the impossibility of sampling uniformly in strictly bounded time from a set $\mathcal{C}_i$ of whose cardinality is not a power of 2. Note that we have to deal with sampling failures in protocols. For simplicity, we may simply accept a completeness error of $\mathcal{O}(2^{-\mathsf{poly}(\lambda)})$, and run $M(1^{\mathsf{poly}(\lambda)})$ to sample with suitable precision.

The notion of reverse sampleable distributions encodes a lack of secret information in a distribution *when sampled using* M.[5] This is what we will need for public-coin protocols, transparent setups, or for programming verifier coins and random oracles during knowledge extraction. Concretely, for reverse sampleable M, we can "program" or "explain" a (predetermined) outcome c in an (almost) perfect way by sampling using the reverse sampler to find suitable coins r so that $c_i = M_i(1^\lambda; \mathsf{RO}(r))$.

As we also deal with *expected* polynomial time algorithms (for knowledge extraction), one must be careful since even a tiny change in distribution can immensely affect the runtime in general. However, a closer inspection shows that $\rho := \rho_{\sup}(p(\,\cdot\,)/U_{\mathcal{C}}) = \max_{c\in\mathcal{C}} p(c)/(1/|\mathcal{C}|) = \max_{x\in\mathcal{C}} p(c)|\mathcal{C}| \leq 1 + 2^{-\mathsf{poly}(\lambda)}$, where $p(c)$ is the probability that $M(1^\lambda)$ outputs c. From this, it easily follows that for any function $T\colon \mathcal{C} \cup \{\bot\} \to \mathbb{R}_{\geq 0}$, we have $\mathbb{E}[T(C)] \leq \rho \cdot \mathbb{E}[T(C')]$ where C is uniformly distributed in $\mathcal{C}$ and $C' = M(1^\lambda)$ and we assume $\mathbb{E}[T(C') \mid C' = \bot] \leq \mathbb{E}[T(C)]$. The latter requirement on T is natural when T is runtime of an algorithm which aborts on sampling $\bot$.

[5] Note that sampling algorithm M is of central importance, even though it is often left implicit. For example, sampling a group element by interpreting a random bitstring as a group element has no secrets, whereas exponentiating the group generator by a random exponent does contain secrets. More generally, if $(x,r) \mapsto M(1^\eta, x; r)$ is one-way, then M cannot be reverse sampleable.

Many distributions of interest satisfy our definition of reverse sampleable.

Example 2.3.7 (Sampling from $\{1, \ldots, n\}$). Choosing uniformly from $\{0, \ldots, n-1\}$ or any set $\mathcal{Y}$ with an efficient *bijection* $\iota\colon \mathcal{X} \to \mathcal{Y}$ is efficiently reverse sampleable w.r.t. the uniform distribution. This is an immediate consequence from rejection sampling. For example, let M choose c uniformly from $\{0,1\}^{\lceil \log(n) \rceil} \cong \{0, \ldots 2^{\lceil \log(n) \rceil} - 1\}$ until $c < n$ or η tries are exceeded, in which case M outputs $\perp$. Clearly, this efficiently samples $U_{\{0,\ldots,n-1\}}$ according to Definition 2.3.6. Moreover, M is reverse sampleable as follows: First, $M^\dagger(1^\eta, y)$ runs $M(1^\eta)$ to sample some output r_t after t tries, with random choices $(r_1, \ldots, r_t) \in \{0, \ldots 2^{\lceil \log(n) \rceil} - 1\}^t$ (or $\perp$). Then $M^\dagger$ outputs $(r_1, \ldots, r_{t-1}, y)$ (or $\perp$ if $M(1^\eta)$ returned $\perp$).

Example 2.3.8 (Efficiently sampling fat subsets). Suppose $\mathcal{Y} \subset \mathcal{Z}$ is a subset of relative size $\rho = \#\mathcal{Y}/\#\mathcal{Z}$ and that $\mathcal{Y}$ is efficiently recognizable and the uniform distribution $U_{\mathcal{Z}}$ is efficiently sampleable and reverse sampleable (in the security parameter λ, i.e. as a family over $\mathcal{X} = \mathbb{N}$). Suppose furthermore that $\rho \geq 1/\mathsf{poly}(\lambda)$ for security parameter λ. Then the uniform distribution $U_{\mathcal{Y}}$ on $\mathcal{Y}$ is efficiently sampleable and reverse sampleable, e.g. as follows: Use repeated trials similar to Example 2.3.7 to sample $U_{\mathcal{Y}}$ with sampling parameter η, e.g. run at least $n = \rho^{-1}(\lambda) \cdot \eta$ trials before giving up. Reverse sampling is also analogous to Example 2.3.7.

Examples 2.3.7 and 2.3.8 already cover a lot of cases, for example reverse sampling in $\mathbb{Z}_n$ for any $n \in \mathbb{N}$, or in $\mathbb{Z}_p^\times$ for prime p, or in the subset $\mathbb{P} \cap \{2^n, \ldots, 2^n - 1\}$ of n-bit primes. By minor variations, one covers many cryptographic groups, such as quadratic residues modulo n, or many elliptic curves which map bijectively to fat subsets (e.g. by mapping to the x coordinate in $\mathbb{Z}_p$ and a bit which specifies which of the (at most) 2 possible y coordinates is chosen). However, there are cryptographic groups of interest where the (im)possibility of reverse sampling is still an open question (at the time of writing), e.g. class groups of imaginary quadratic orders (cf. Appendix A.1.2).

Aside 2.3.9. Many definitional variants of Definition 2.3.6 are possible. One example is to consider $\Delta(X_{\eta,x}, D_x) \leq 2^{-\eta}$ in Definition 2.3.6 instead of perfect-unless-$\perp$. Moreover, in most applications, for reverse sampling it is sufficent if $M^\dagger$ produces almost uniform r in $\{r \in \{0,1\}^{\mathsf{poly}(\eta,|x|)} \mid M(1^\eta, x; r) = y\}$. Lastly, the requirements for efficient (reverse) sampling can be weakened to average case requirements, since the x (resp. x and y) often follow a distribution.

2.3.2.2. Public-Coin and Transparent Setup

As noted, reverse sampleable distribution have “no secrets” and can thus be used to give a (generalized) definition of public-coin verifiers and transparent setup that more naturally corresponds to their intuitive meaning and real-world instantiations.

Definition 2.3.10 (Public-coin). An interactive proof system $(\mathsf{GenCRS}, \mathsf{P}, \mathsf{V})$ is **(generalized) public-coin** if V’s message in the $2i$-th move, called the *challenge* c_i is sampled via a sampling algorithm $M(1^\eta, tr_i)$ where tr_i is the transcript up to the $2i$-th move. Moreover, M must be reverse sampleable. Without loss of generality, the final output b of a public-coin verifier is $b = \mathsf{Verify}(x, tr)$ for a PPT algorithm Verify given the full transcript tr.

Aside 2.3.11. An alternative to public-coin proof systems is to simply define security notions where the adversary is given the random coins of the verifier (and put no explicit restrictions on the verifier). This is the most liberal notion we are aware of. It covers pathological examples, e.g. when the protocol ignores a part of the challenge which the verifier samples non-reversibly. However, we are not aware of any non-pathological protocol of interest which not already satisfies our stricter notion of public-coin.

Next, we define transparent setups.

Definition 2.3.12 (Transparent setup). An interactive proof $(\mathsf{GenCRS}, \mathsf{P}, \mathsf{V})$ has **(generalized) transparent setup** if GenCRS is reverse sampleable.

Since the CRS *crs* of a transparent setup can w.l.o.g. be the (uniformly) random coins of GenCRS, one also calls such a CRS a **uniform random string (URS)** and its non-transparent counterpart a **structured random string (SRS)**. Clearly, it is easier to generate a URS than an SRS in practice, e.g. by using "nothing-up-my-sleeve" methods, such as using the digits of π.

2.3.3. Zero-Knowledge

We define honest-verifier zero-knowledge for public-coin proof systems. Full-fledged zero-knowledge is only considered in Chapter 5 and defined there.

Definition 2.3.13 ((Non-Abort) (S)HVZK). A simulator Sim for a public coin proof system $(\mathsf{GenCRS}, \mathsf{P}, \mathsf{V})$ for $\mathcal{R}$ is a PPT algorithm with input a statement $\mathbb{x}$ for which $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$ and implicit inputs 1^λ, *pp*, *crs*, and output a transcript *tr*. Let $\mathcal{A}$ be a stateful algorithm and let

$$\mathsf{Real}_{\mathcal{A}}(\lambda) = \Pr\begin{bmatrix} pp \leftarrow \mathsf{GenPP}(1^\lambda); crs \leftarrow \mathsf{GenCRS}(pp); \\ (\mathbb{x}, \mathbb{w}) \leftarrow \mathcal{A}(pp, crs); \\ tr \leftarrow \langle \mathsf{P}(pp, crs, \mathbb{x}, \mathbb{w}), \mathsf{V}(pp, crs, \mathbb{x}) \rangle; \\ b \leftarrow \mathcal{A}(tr) : \ b \wedge \mathcal{R}(\mathbb{x}; \mathbb{w}) = 1 \end{bmatrix}$$

$$\mathsf{Ideal}_{\mathcal{A}}(\lambda) = \Pr\begin{bmatrix} pp \leftarrow \mathsf{GenPP}(1^\lambda); crs \leftarrow \mathsf{GenCRS}(pp); \\ (\mathbb{x}, \mathbb{w}) \leftarrow \mathcal{A}(pp, crs); tr \leftarrow \mathsf{Sim}(pp, crs, \mathbb{x}); \\ b \leftarrow \mathcal{A}(tr) : \ b \wedge \mathcal{R}(\mathbb{x}; \mathbb{w}) = 1 \end{bmatrix}$$

Define the advantage of $\mathcal{A}$ by $\mathsf{Adv}^{\mathrm{hvzk}}_{\mathcal{A},\mathsf{P},\mathsf{V}}(\lambda) = \mathsf{Real}_{\mathcal{A}}(\lambda) - \mathsf{Ideal}_{\mathcal{A}}(\lambda)$. Then Sim (and by extension $(\mathsf{GenCRS}, \mathsf{P}, \mathsf{V})$) is **honest verifier zero-knowledge** with **simulation error** $\delta_{\mathrm{sim}} = \delta_{\mathrm{sim}}(\lambda)$, if for all PPT $\mathcal{A}$ there exists a negligible function negl such that $\mathsf{Adv}^{\mathrm{hvzk}}_{\mathcal{A},\mathsf{P},\mathsf{V}} \leq \delta_{\mathrm{sim}} + \mathsf{negl}$. It is **statistical** HVZK by $\mathsf{Adv}^{\mathrm{hvzk}}_{\mathcal{A},\mathsf{P},\mathsf{V}} \leq \delta_{\mathrm{sim}}$ for all (even unbounded) $\mathcal{A}$.

If in the real and ideal experiments, $\mathcal{A}$ is allowed to *prescribe* the challenges that the honest verifier (or simulator) will use, then we denote the advantage by $\mathsf{Adv}^{\mathrm{shvzk}}_{\mathcal{A},\mathsf{P},\mathsf{V}}$ and call the resulting notion ***special** honest verifier zero-knowledge (SHVZK)*.

The simulator is **non-abort** (S)HVZK, if it satisfies the weaker requirement, that simulated transcripts and real non-aborting transcripts are indistinguishable. Formally, use the modified $\mathsf{Real}_{\mathcal{A}}$, where the transcript *tr* is replaced by $\perp$ if the honest prover aborts, to define the advantage $\mathsf{Adv}^{\mathrm{na\text{-}hvzk}}_{\mathcal{A},\mathsf{P},\mathsf{V}}$ (resp. $\mathsf{Adv}^{\mathrm{na\text{-}shvzk}}_{\mathcal{A},\mathsf{P},\mathsf{V}}$).

Remark 2.3.14 (Non-abort (S)HVZK to ordinary (S)HVZK). The protocols in Chapter 3 are only *non-abort* (S)HVZK. If "standard" (S)HVZK is needed, it can be obtained via well-known transformations. For example, via committing to those messages which, in case of failed masking, the simulator could not compute backwards. If these messages have enough entropy, suitable hashing (which is collision resistant and hides high-entropy preimages) suffices.

Remark 2.3.15. We do not define or use full-fledged zero-knowledge here, but note that it is easily achieved if we are allowed to modify the setup. For example, by using an equivocal (Blum) coin-toss to choose the challenges. There is a catch, namely, one might have to change the witness relation, relax to computational (knowledge) soundness, or use relaxed soundness (as discussed in Section 2.3.4) to compensate the possibility of breaking the commitment.

2.3.4. Knowledge Soundness

There is a plethora of sensible, useful, and closely related definitions of knowledge soundness. We present one which is intuitive and easy to use in our setting. Moreover, we concentrate on *(statistical) knowledge errors* to have an explicit measure of soundness (instead of requiring asymptotically negligible errors). This allows us to analyze the behaviour of computational security parameter and knowledge error almost independently. Moreover, we consider a relation for knowledge soundness $\mathcal{R}_{\mathsf{Ext}}$, which may differ from the relation $\mathcal{R}$ for correctness.

Definition 2.3.16 (Knowledge Error). Let $(\mathsf{GenCRS}, \mathsf{P}, \mathsf{V})$ be a public-coin interactive proof system for NP-relation $\mathcal{R}$ with **relaxed (knowledge) soundness relation** $\mathcal{R}_{\mathsf{Ext}}$. Let Ext be an *expected* polynomial time oracle algorithm (with oracle steps counted as one step) with implicit inputs 1^λ, pp, crs, explicit inputs $\mathbb{x}$, tr, and output $\mathbb{w}$, a (purported) witness or $\perp$. We call Ext a **(black-box) extractor**. Let $\mathcal{A}$ be a probabilistic algorithm and P^* be a deterministic algorithm.

$$\mathsf{Real}_{\mathcal{A}}(\lambda) = \Pr\left[\begin{array}{c} pp \leftarrow \mathsf{GenPP}(1^\lambda); crs \leftarrow \mathsf{GenCRS}(pp); \\ (\mathbb{x}, aux) \leftarrow \mathcal{A}(pp, crs); tr \leftarrow \langle \mathsf{P}^*(\mathbb{x}, aux), \mathsf{V}(\mathbb{x})\rangle : \\ \mathsf{Verify}(\mathbb{x}, tr) = 1 \end{array}\right]$$

$$\mathsf{Ideal}_{\mathcal{A}}(\lambda) = \Pr\left[\begin{array}{c} pp \leftarrow \mathsf{GenPP}(1^\lambda); crs \leftarrow \mathsf{GenCRS}(pp); \\ (\mathbb{x}, aux) \leftarrow \mathcal{A}(pp, crs); tr \leftarrow \langle \mathsf{P}^*(\mathbb{x}, aux), \mathsf{V}(\mathbb{x})\rangle; \\ \mathbb{w} \leftarrow \mathsf{Ext}^{\mathsf{P}^*(\mathbb{x}, aux)}(\mathbb{x}, tr) : \\ \mathsf{Verify}(\mathbb{x}, tr) = 1 \land (pp, crs; \mathbb{x}; \mathbb{w}) \in \mathcal{R}_{\mathsf{Ext}} = 1 \end{array}\right]$$

W.l.o.g. Ext sets $\mathbb{w} = \perp$ if $\mathsf{Verify}(\mathbb{x}, tr) \neq 1$. The advantage of $(\mathcal{A}, \mathsf{P}^*)$ is $\mathsf{Adv}^{\mathrm{ke}}_{\mathcal{A},\mathsf{P}^*,\mathsf{V}}(\lambda) = \mathsf{Real}_{\mathcal{A}}(\lambda) - \mathsf{Ideal}_{\mathcal{A}}(\lambda)$.

An extractor Ext has **(statistical) knowledge error** κ, if for any (possibly unbounded) adversary $(\mathcal{A}, \mathsf{P}^*)$, we have $\mathsf{Adv}^{\mathrm{ke}}_{\mathcal{A},\mathsf{P}^*,\mathsf{V}} \leq \kappa$. Such proof systems are called **proofs of knowledge** (or **arguments of knowledge**).[6] If $\mathcal{R}_{\mathsf{Ext}} \neq \mathcal{R}$, i.e. the relation for correctness and knowledge soundness differ, we speak of **relaxed (knowledge) soundness** for relaxed soundness relation $\mathcal{R}_{\mathsf{Ext}}$.

More generally, we allow a **knowledge error function** $\mathsf{fn}_{\mathrm{knw}}$ for extractors, where $\Pr[\mathsf{Ideal}_{\mathcal{A}}] \geq \mathsf{fn}_{\mathrm{knw}}(\Pr[\mathsf{Real}_{\mathcal{A}}])$ must hold for statistical soundness. (The standard definition of knowledge error corresponds to $\mathsf{fn}_{\mathrm{knw}}(\varepsilon) = \varepsilon - \kappa$.)

Notation 2.3.17. When defining ad-hoc (relaxed) knowledge relations, we sometimes use the shorthand notation

$$\text{\reflectbox{E}}\mathbb{w} \colon \mathsf{pred}(\mathbb{x}, \mathbb{w}),$$

[6] We can analogously define the **computational knowledge error** κ. However, we will not need it and it is simpler and more convenient if computational assumptions can instead be put into the relaxed soundness relation, see Remark 2.3.19. For concreteness: Ext has computational knowledge error κ if for every PPT pair $(\mathcal{A}, \mathsf{P}^*)$, there exists a negligible function negl such that $\mathsf{Adv}^{\mathrm{ke}}_{\mathcal{A},\mathsf{P}^*,\mathsf{V}} \leq \kappa + \mathsf{negl}$.

where K is a "knowledge quantifier" and pred is a predicate. This translates into a relation $\mathcal{R}$ via $\mathcal{R} = \{(\mathbb{x}, \mathbb{w}) \mid \mathsf{pred}(\mathbb{x}, \mathbb{w})\}$, and we leave λ, *pp*, *crs* implicit, as usual.

Convention 2.3.18. If it is clear from the context, we say *soundness* instead of *knowledge soundness.* For example, if we describe extractors, consider proofs of knowledge, or discuss properties where only witness extraction makes sense (e.g. openings of commitments).

Definition 2.3.16 assumes *black-box* extraction. Furthermore, knowledge soundness is only defined for public-coin protocols. For general proof systems, one must replace the transcript by the verifier's view and $\mathsf{Verify}(\mathbb{x}, tr)$ by the verifier's output bit b_V. In any case, Definition 2.3.16 (or the described generalization) intuitively ensure that one can extract any accepting conversation, except with probability κ.

Remark 2.3.19 (Advantages of relaxed soundness). The possibility of a *relaxed* witness relation $\mathcal{R}_\mathsf{Ext} \neq \mathcal{R}$ significantly simplifies many theorem statements and allows a relatively clean separation of extraction (which is statistical and even non-asymptotic) and hardness assumptions and security reduction. For example, the relaxed soundness relation $\mathcal{R}_\mathsf{Ext}$ might allow a break of a hard relation (e.g. binding property of commitment broken) as a witness. Clearly, one can construct an adversary against the binding property from such a witness. Hence, (statistical) relaxed knowledge soundness w.r.t. $\mathcal{R}_\mathsf{Ext}$ implies computational soundness for $\mathcal{R}$ (Footnote 6). This example is typical for the definition of $\mathcal{R}_\mathsf{Ext}$. Indeed, in Chapters 3 and 4, $\mathcal{R}_\mathsf{Ext}$ usually has the form "either $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$ or $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}_\mathsf{hard}$", where $\mathcal{R}_\mathsf{hard}$ is a hard relation which captures, e.g. a hash collision or a binding break for a commitment.

Remark 2.3.20 (Definitional variations). Many variations of knowledge soundness can be defined. Our definition is relatively strong and tailored to our setting of public-coin proof systems. In fact, our actual constructions of extractors will satisfy even stricter properties. In the other direction, the notion of *witness-extended emulation* [Lin03; GI08] is less restrictive on extraction strategies, and it is easily seen that witness-extended emulation is implied by Definition 2.3.16.

Most of our proof systems crucially rely on either k-special soundness, or very similar techniques, to ensure knowledge soundness. That is, given k "related transcripts", one can reconstruct a witness. While not all of our proof systems are strictly k-special sound, e.g. in Chapter 3 the protocol $\mathsf{Sharp}_\mathsf{SO}^\mathsf{Po}$ is a 5-move protocol (which does not satisfy the tree-of-transcripts generalization of special soundness either), the same techniques which are used for special sound protocol and tree-finding are applicable. Indeed, these ideas are useful in many contexts.

Thus, we now define (multi-round) special soundness, and then consider tree-finding (i.e. obtaining related transcripts) and explain how putting this together gives a knowledge extractor.

2.4. Knowledge Extraction and Special Soundness

In this section, we discuss knowledge extractor based on special soundness and split into a tree-finder and a tree-extractor.

2.5. Special Soundness

The standard definition of special soundness pertains 3-move public-coin proof systems (also called Σ-**protocols**). So let (GenCRS, P, V) be such a proof system for $\mathcal{R}$. It is k-special sound for some $k \in \mathbb{N}$ if given k related transcripts tr_i with same first message a but distinct challenges, one can efficiently compute a witness $\mathbb{w}$ for the statement $\mathbb{x}$. More concretely, there is a PPT algorithm that given $\mathbb{x}$ and a set $\{(a, \gamma_i, z_i)\}_{i=1}^{n}$ of transcripts where $tr_i = (pp, crs, \mathbb{x}, a, \gamma_i, z_i)$ outputs $\mathbb{w}$ such that $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$.

In multi-round protocols, where ℓ challenges are issued, the natural generalization is a tree of transcripts, where layer i nodes have k_{i+1} children, starting with layer 0, the root, with k_1 children and ending with the leaves in layer ℓ (without children). This is illustrated in Fig. 2.1. The formal definition follows.

Definition 2.5.1 (Tree of Transcripts). Let $\Pi = (\mathsf{GenCRS}, \mathsf{P}, \mathsf{V})$ be a public-coin proof system for NP-relation $\mathcal{R}$. A $(k_1, \dots, k_\ell)$-**tree** *tree* **of transcripts** is defined as follows: It is a set of $K = \prod_{i=1}^{\ell} k_i$ transcripts $tree = \{(pp, crs, \mathbb{x}; a_0, \gamma_1, a_1, \dots, \gamma_\ell, a_\ell)\}$ where each a_i a message from the prover, such that *tree* has a labelled $(k_1, \dots, k_\ell)$-tree structure as follows. The root is $(pp, crs, \mathbb{x}; a_0)$. It has k_1 edges labelled with challenges in $\mathcal{C}_1$. The next layer has nodes labelled with messages a_1 and k_2 edges labelled with challenges from $\mathcal{C}_2$ and so on. Following a path from the root to a leaf induces a transcript $(pp, crs, \mathbb{x}; a_0, \gamma_1, a_1, \dots, \gamma_\ell, a_\ell)$ in *tree*. A tree is **valid** if all (inner) nodes have distinctly labelled edges (i.e. correspond to distinct challenges).

Another analogous definition $(k_1, \dots, k_\ell)$-**trees of challenges**. Here, one considers a (probabilistic) algorithm A which takes as input a sequence of challenges $(\gamma_1, \dots, \gamma_\ell)$ and outputs 0 or 1. Only the root is labelled with A's random tape r.

We also write $\vec{k}$-tree for $\vec{k} = (k_1, \dots, k_\ell)$. Observe that layer i has $\prod_{j=i+1}^{\ell} k_{\ell_i}$ children (starting with layer 0 at the root) and overall a $\vec{k}$-tree consists of $k = \prod_{i=1}^{\ell} k_i$ transcripts.

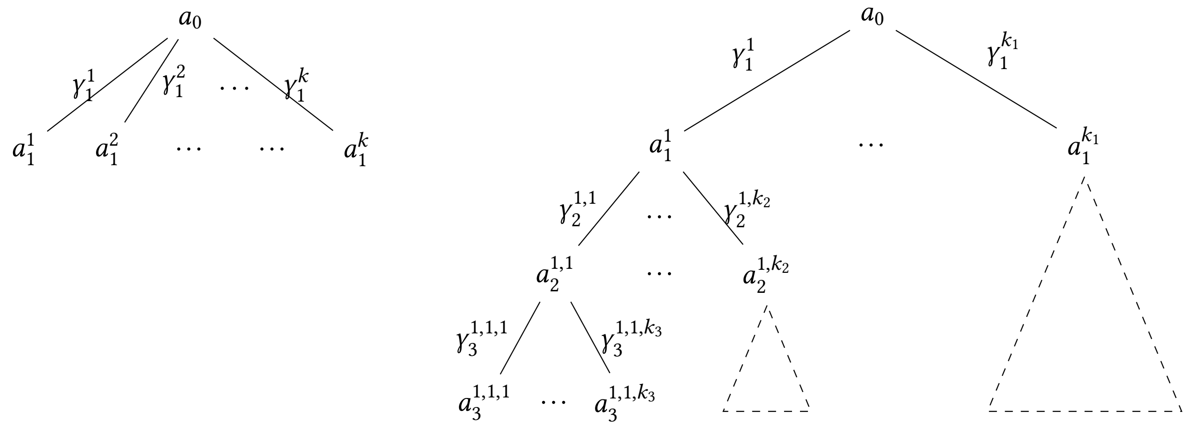


Figure 2.1.: Trees of transcripts. Left: k related transcripts, i.e. a k-tree for a Σ-protocol. Right: A schematic $(k_1, \dots, k_3)$-tree.

Definition 2.5.2 (Special Soundness). Let $\Pi = (\mathsf{GenCRS}, \mathsf{P}, \mathsf{V})$ be a public-coin proof system for NP-relation $\mathcal{R}$. Suppose that the prover moves first and that ℓ challenges are sent by the verifier (i.e. there is a total of $2\ell + 1$ moves). Suppose furthermore that in the i-th move of the verifier, the challenge is chosen uniformly from a challenge set $\mathcal{C}_i$.

We call Π special sound with **tree extractor** TreeExt for (potentially relaxed soundness) relation $\mathcal{R}_{\mathsf{Ext}}$ if TreeExt is a PPT algorithm which given a valid tree *tree* for statement $\mathbb{x}$ outputs a witness $\mathbb{w}$ in $\mathcal{R}_{\mathsf{Ext}}$.

Convention 2.5.3. Unless stated otherwise, whenever we define a tree extractor, we only consider **valid** trees. (Observe that validity is efficiently testable.)

The tree structure is essentially a consequence of sequentially composing Σ-protocols by proving that a response is accepting, instead of sending that response directly. Since such proofs can be smaller than sending the witness, it allows for (recursively) compressing the communication. This is exploited in Chapter 4. Unsurprisingly, the idea of special soundness, namely that a number of transcripts allow extracting a witness, is a typical approach for achieving knowledge soundness even in settings which do not strictly fall into our definition of special soundness.

Now, we turn to the question of *finding* valid $\vec{k}$-trees.

2.5.1. Tree-Finders

Now, we exemplify some tree-finding strategies. First, we define a generic notion of tree-finder

Definition 2.5.4. Let A be an algorithm which takes as input a ℓ-tuple $(\gamma_1, \dots, \gamma_\ell) \in \mathcal{C}_1 \times \dots, \times \mathcal{C}_\ell$ of challenges and outputs 0 or 1. A $(k_1, \dots, k_\ell)$-tree-finder $\mathsf{TreeFind}$ is given black-box access to A and outputs a *valid* $(k_1, \dots, k_\ell)$-tree of challenges *tree* or $\bot$. We say that $\mathsf{TreeFind}$ admits **knowledge error function** $\mathsf{fn}_{\mathrm{knw}}$ if for all such A we have

$$\Pr[\mathsf{TreeFind}^{\mathsf{A}} \neq \bot] \geq \mathsf{fn}_{\mathrm{knw}}(\Pr_{\vec{\gamma},r}[\mathsf{A}(\gamma_1, \dots, \gamma_\ell; r)]).$$

We say $\mathsf{TreeFind}$ has **knowledger error** κ if knowledge error function $\mathsf{fn}_{\mathrm{knw}}(\varepsilon) = \varepsilon - \kappa$ is admissible.

We consider a general knowledge error *function* in Definition 2.5.4 to capture different tree-finding strategies in one sweep. Moreover, we abstract away proof systems altogether and consider any algorithm A which takes as input challenges and outputs a bit, which should be viewed as the verifier's output bit (given the challenges). For public-coin proof systems, this abstraction works well. Moreover, it is easy to see that the knowledge error of $\mathsf{TreeFind}$ can be checked solely by considering deterministic A.

Corollary 2.5.5. *In Definition 2.5.4, it suffices w.l.o.g. to check the knowledge error for deterministic* A*. The same holds for any* convex *knowledge error functions.*

Proof. By definition of black-box rewinding access, the randomness of A is chosen uniformly and then fixed (during any rewinds). Thus, we have

$$\begin{aligned}
\Pr[\mathsf{TreeFind}^{\mathsf{A}(\,\cdot\,)} \neq \bot] &= \mathbb{E}_r[\Pr[\mathsf{TreeFind}^{\mathsf{A}(\,\cdot\,;r)} \neq \bot]] \\
&\geq \mathbb{E}_r[\mathsf{fn}_{\mathrm{knw}}(\Pr_{\vec{\gamma}}[\mathsf{A}(\vec{\gamma}; r) = 1])] \\
&\geq \mathsf{fn}_{\mathrm{knw}}(\mathbb{E}_r[\Pr_{\vec{\gamma}}[\mathsf{A}(\vec{\gamma}; r) = 1]])^{\cdot} \\
&= \mathsf{fn}_{\mathrm{knw}}(\Pr_{\vec{\gamma},r}[\mathsf{A}(\vec{\gamma}; r) = 1])
\end{aligned}$$

where the first step is basic probability theory, the second follows from the claim for the now *deterministic* $\mathsf{A}(\,\cdot\,; r)$, the third is a consequence of Jensen's inequality (since $\mathsf{fn}_{\mathrm{knw}}$ is convex), and the last is again basic probability theory. □

Note that Definition 2.5.4 is also non-asymptotic, which further simplifies working with it. Nevertheless, we have following straightforward fact.

Corollary 2.5.6. *Let* $(\mathsf{GenCRS}, \mathsf{P}, \mathsf{V})$ *be a* $(k_1, \dots, k_\ell)$*-special sound public-coin proof system with tree-extractor* $\mathsf{TreeExt}$. *Suppose* $\mathsf{TreeFind}$ *is a* $(k_1, \dots, k_\ell)$*-tree-finder that works by first querying* A *on a uniformly random challenge sequence* $(\gamma_1, \dots, \gamma_\ell) \in \mathcal{C}_1 \times \dots \times \mathcal{C}_\ell$. *Suppose furthermore that* $\mathsf{TreeFind}$ *has knowledge error* κ. *Then we obtain an extractor* Ext *with knowledge error* κ *by running* $\mathsf{TreeFind}$ *and then* $\mathsf{TreeExt}$. *An analogous claim same holds for the knowledge error function.*

Proof. The definition of $\mathsf{A}(\gamma_1, \dots, \gamma_\ell)$ for $\mathsf{TreeFind}$, provided as a black-box oracle by $\mathsf{Ext}(\mathbb{x}, aux)$, is simply an execution of $\mathsf{P}^*(\mathbb{x}, aux, tr)$ with challenges $(\gamma_1, \dots, \gamma_\ell)$ and the output being the verifier's verdict, i.e.

$$\mathsf{A}(\gamma_1, \dots, \gamma_\ell) \coloneqq \mathsf{out}_{\mathsf{V}}\langle \mathsf{P}^*(\mathbb{x}, aux, tr), \mathsf{V}(\mathbb{x}; \gamma_1, \dots, \gamma_\ell)\rangle.$$

There is a minor subtlety, namely that Ext takes as input the transcript *tr* of the prior honest interaction. However, since $\mathsf{TreeFind}$ makes a first query with uniformly random challenges sequence $(\gamma_1, \dots, \gamma_\ell)$, Ext can program the randomness of $\mathsf{TreeFind}$ to start with this challenge sequence (without affecting the distribution of $\mathsf{TreeFind}$). With this, the claim follows. □

With Corollary 2.5.6, we have reduced the construction of efficient extractors to the construction of efficient tree-finders and tree-extractors. As the latter are often trivially efficient, the runtime of an extractor with this blueprint is dominated by the runtime of the tree-finder.

2.5.1.1. Basic Extraction Strategies

Now, we describe two basic extraction strategies for Σ-protocols, i.e. a single challenge ($\ell = 1$), i.e. a k-tree, i.e. k related transcripts.

Random Challenges with Replacement. The most straightforward approach to k-tree-finding is the following algorithm, which we call $\mathsf{TreeFind}_{\mathsf{Geo}}$. Let $\mathcal{C}$ be the challenge set and A be a deterministic algorithm (as in Definition 2.5.4).

1. Pick $\gamma_1 \xleftarrow{\$} \mathcal{C}$ uniformly.
2. If $\mathsf{A}(\gamma_1) = 0$ return $\bot$. Else let $i = 2$.
3. Repeat until $i > k$
   - Pick $\gamma_i \leftarrow \mathcal{C}$ uniformly.
   - If $\mathsf{A}(\gamma_i) = 1$, increase i to $i + 1$.
4. If $\gamma_1, \dots, \gamma_k$ are distinct, output $(\gamma_1, \dots, \gamma_k)$, else output $\bot$.

The *runtime analysis* of $\mathsf{TreeFind}_{\mathsf{Geo}}$ in terms of queries to A is quite simple: Let $\varepsilon = \Pr_{\vec{\gamma}}[\mathsf{A}(\vec{\gamma}) = 1]$. Then with probability $1 - \varepsilon$, TreeFind immediately returns $\bot$. With probability ε, it enters the loop. For each i, the number of trials until a success follows a geometric distribution $\mathrm{Geo}(\varepsilon)$. Recall that $\mathrm{Geo}(p)$ has expectation $1/p$. Let Q be the number of total queries TreeFind makes and Q_i be the number of queries until an accepting γ_i is found. Then, we find the following:

$$\mathbb{E}[Q] = 1 + \varepsilon \cdot (\sum_{i=2}^{k} \mathbb{E}[Q_i]) = 1 + \varepsilon \cdot (k-1) \cdot \frac{1}{\varepsilon} = k$$

Thus, the expected number of queries is exactly k, which is essentially optimal.

The success analysis gives a less convincing result, due to possible collisions of challenges. Indeed, if A accepts (i.e. outputs 1) for exactly m out of $N = \#\mathcal{C}$ challenges, i.e. accepts with probability $\varepsilon = \frac{m}{N}$, then we find a success probability of only

$$m/N \cdot \prod_{i=0}^{\ell-1} \frac{m-i}{m} = \varepsilon \cdot \frac{m!}{(m-k)!m^k}$$

with $m = k$ yielding a worst case of $\varepsilon \cdot \frac{k!}{k^k} \approx \varepsilon \cdot \sqrt{2\pi k} \cdot e^{-k}$. Thus, knowledge error $\kappa = k/N$ cannot be (efficiently) achieved with this strategy.

Sampling challenges without replacement. It is clear that for *deterministic* A sampling challenges *with replacement* is wasteful, because the outcome of a retry can be predicted. Indeed, for any fixed number $n < N = \#\mathcal{C}$, sampling without replacement actually maximizes the probability of success (and there is never a collision). Thus, an obvious question is whether or not sampling without replacement is also efficient. This k-tree-finder was studied in [ACK21] where it is shown that its expected time is still k, hence essentially optimal.

Now, we recall this tree-finder, denoted $\mathsf{TreeFind}_{\mathsf{NHG}}$, in more detail. It works as follows:

1. Pick $\gamma_1 \xleftarrow{\$} \mathcal{C}$ uniformly.
2. If $\mathsf{A}(\gamma_1) = 0$ return $\bot$. Else let $i = 2$.
3. Repeat until $i > k$ or all of $\mathcal{C}$ was exhausted.
   - Pick γ_i uniformly from $\mathcal{C}$ without replacement.
   - If $\mathsf{A}(\gamma_i) = 1$, increase i to $i + 1$.
4. If $i \leq k$, return $\bot$. Else return $(\gamma_1, \dots, \gamma_k)$.

The runtime distribution of the looped part now follows a *negative hypergeometric distribution*, and the expected number of queries TreeFind makes is (bounded by) k, see [ACK21] for details. Let $\varepsilon = \Pr_{\vec{\gamma}}[\mathsf{A}(\vec{\gamma}) = 1]$. Then the success probability of $\mathsf{TreeFind}_{\mathsf{NHG}}$ is ε if $\varepsilon \geq k/N$ for $N = \#\mathcal{C}$ and 0 otherwise. Since this knowledge error function is not convex, we use following convex lower bound:

$$\mathsf{fn}_{\mathrm{knw}}(\varepsilon) = \frac{\varepsilon - \kappa}{1 - \kappa}$$

for knowledger error $\kappa = k/N$.

2.5.2. Recursive Tree-Finding

Given (a uniformly described family of) basic k-tree-finders $\mathsf{TreeFind}_k$ for $k \in \mathbb{N}$, we can construct a $(k_1, \dots, k_\ell)$-tree-finder in a straightforward manner. Namely, by a form of "sequential composition" which we sketch here. For this, we implicitly extend the definition of A from Definition 2.5.4 as follows: A takes an input c and outputs a pair (b, s) of a success bit b and some auxiliary string s (which is not relevant for success).

Let A be as above. Define $\mathsf{A}_\ell(c_1, \dots, c_\ell) \coloneqq \mathsf{A}(c_1, \dots, c_\ell)$, i.e. $\mathsf{A}_\ell = \mathsf{A}$. For $i = 1, \dots, \ell$, define intermediate algorithms A_{i-1} with inputs $(c_1, \dots, c_{i-1})$ as follows: The algorithm $\mathsf{A}_{i-1}(c_1, \dots, c_{i-1})$ uses $\mathsf{TreeFind}_{k_i}^{\mathsf{A}_i(c_1,\dots,c_{i-1},\cdot)}$ to obtain a tree of challenges with auxiliary information. That is, if $\mathsf{TreeFind}_{k_i}^{\mathsf{A}_i(c_1,\dots,c_{i-1},\cdot)}$

succeeds, it outputs a list $z = ((\gamma_1, s_1), \ldots, (\gamma_{k_i}, s_i))$, where $\mathsf{A}_i(c_1, \ldots, c_{i-1}, \gamma_j) = (1, s_j)$. In this case, A_{i-1} outputs $(1, z)$. If $\mathsf{TreeFind}_{k_i}^{\mathsf{A}_i(c_1,\ldots,c_{i-1},\,\cdot\,)}$ fails, it outputs $\bot$, and $\mathsf{A}_{i-1}(c_1, \ldots, c_{i-1})$ outputs $(0, \bot)$.

Observe that, recursively, the auxiliary string s output by $\mathsf{A}_{i-1}(c_1, \ldots, c_{i-1})$ has the form of a $(k_i, \ldots, k_\ell)$-subtree of challenges (with prefix $(c_1, \ldots, c_{i-1})$), until $\mathsf{A}_0 = \mathsf{TreeFind}_{k_1}^{\mathsf{A}_1(\,\cdot\,)}$ actually outputs a full $(k_1, \ldots, k_\ell)$-tree of challenges for A in the auxiliary string s on success $(1, s)$ (or $(0, \bot)$ on failure).

Define success probabilities $\varepsilon_i = \Pr_{\gamma_1,\ldots,\gamma_i,r}[\mathsf{A}_i(\gamma_1, \ldots, \gamma_i; r)]$ and observe that $\varepsilon_\ell = \Pr_{\gamma_1,\ldots,\gamma_i,r}[\mathsf{A}_i(\gamma_1, \ldots, \gamma_\ell; r)] = \varepsilon$. By definition of knowledge error functions, we find

$$\varepsilon_{i-1} \geq \mathsf{fn}_{\mathrm{knw},i}(\varepsilon_i).$$

By induction, for $\mathsf{TreeFind}_{\mathsf{NHG}}$ this simplifies to

$$\varepsilon_{i-1} \geq \frac{\varepsilon_\ell - \kappa}{1 - \kappa}$$

where the total knowledge error κ is obtained from the i-th round knowledge errors $\kappa_i = k_i/\#\mathcal{C}_i$ from the formula

$$\kappa = 1 - \prod_{i=1}^{\ell}(1 - \kappa_i) \leq \sum_{i=1}^{\ell} \kappa_i.$$

Intuitively, it should hold that the expected number of queries this recursive construction of a tree-finder makes is bounded by the tree size

$$K = \prod_{i=1}^{\ell} k_i$$

because layer i multiplies the expectation by k_i, and this intuition is true. However, there is a subtlety in the analysis: Our efficiency bounds for queries of $\mathsf{TreeFind}_k$ to A are given *in expectation.* Consequently, A_i for $i < \ell$ make an *expected* number of queries to the underlying A as well. It is well-known that, in general, expected time oracles do not compose well. However, we consider very specific algorithms, and indeed, it can be shown that the basic tree-finders $\mathsf{TreeFind}_{\mathrm{Geo}}$ (resp. $\mathsf{TreeFind}_{\mathsf{NHG}}$) are rewinding strategies with *runtime tightness* k, cf. Section 5.6.1.1 for a formal definition. Roughly, ignoring the steps made in $\mathsf{TreeFind}_{\mathrm{Geo}}$ (resp. $\mathsf{TreeFind}_{\mathsf{NHG}}$) and counting only those in A, $\mathsf{time}_{\mathsf{A}}(\mathsf{TreeFind}^{\mathsf{A}}) \leq k \cdot \mathsf{time}_{\mathsf{A}}(\mathsf{A}(\gamma))$ for uniformly random $\gamma \xleftarrow{\$} \mathcal{C}$. In other words, despite the subtleties of counting *in expectation*, each application of $\mathsf{TreeFind}_{k_i}$ to A_{i+1} increases the *expected* number of queries to A by a factor of at most k_i if $\mathsf{TreeFind}_{\mathrm{Geo}}$ or $\mathsf{TreeFind}_{\mathsf{NHG}}$ is used. Consequently, the expected number of queries to A is bounded by $K = \prod_{i=1}^{\ell} k_i$.

Overall, we find that a recursive composition of $\mathsf{TreeFind}_{\mathsf{NHG}}$ leads to essentially[7] optimal expected runtime and knowledge error for $(k_1, \ldots, k_\ell)$-tree-finders. For an explicit analysis of runtime and success probability of $\mathsf{TreeFind}_{\mathsf{NHG}}$, refer to [ACK21; AFK22].

[7] For $\mathsf{TreeFind}_{\mathrm{Geo}}$, one can abort upon a collision. For $\mathsf{TreeFind}_{\mathsf{NHG}}$, it is possible to reorder the recursive queries and return $\bot$ earlier, if it occurs. Thus, the runtimes are not minimal for *any* prover (but they are minimal for honest provers).

Part II.

Contents

3. Sharp: <u>**Sh**</u>**ort Rel**<u>**a**</u>**xed** <u>**R**</u>**ange** <u>**P**</u>**roofs**

This chapter is based on [CGKR22a; CGKR22b]. It is taken verbatim from [CGKR22b] with minor changes.

My primary contributions in this work are the batch proof of shortness (PoSO), in particular the core lemma. Moreover, I devised and proved security of the augmentation of the scheme with hidden order groups, though the initial suggestion of using hidden order groups is due to Geoffroy Couteau. Minor contributions include separation of proof of shortness and square decomposition into two mostly separate steps, switching to relaxed soundness (instead of using "bounded integer commitment schemes" [CKLR21b]), and exploiting prior knowledge to obtain standard soundness in applications to (updatable) anonymous credentials.

The application of lattice-based techniques to further improve the efficiency of the 3-square decomposition proof and also enable simple and efficient group-switching are primarily due to Michael Reichle. He also worked out an exemplary application of our techniques to anonymous transactions, of which we only include the short overview in Section 3.7.3, see [CGKR22b] the details. The implementations and benchmarks are by Dahmun Goudarzi.

3.1. Introduction

Zero-Knowledge Proofs and Range Proofs. Zero-knowledge proofs, introduced in the seminal work of Goldwasser, Micali, and Rackoff [GMR89], allow a prover to convince a verifier of the truth of a statement while concealing all other information. This makes them an important tool in theory and practice. Efficient constructions are now known for a variety of NP-languages, and are routinely used in real-world applications. An example of particular interest is range proofs, which are zero-knowledge proofs for demonstrating that a secret value (committed or encrypted) belongs to a public range. Range proofs are a core component in numerous applications, such as anonymous credentials [Cha90], e-voting [Gro05], or e-cash [CHL05], and have been introduced recently in some popular anonymous cryptocurrencies (see [Zca; Mon; BAZB20]).

Range Proofs. Many range proofs which have been constructed in the past can be categorized in two main paradigms:

(1) Range proofs based on n-ary decomposition [CCs08; Gro11], where one proves a statement of the form $x \in [0, n^\ell)$ by committing to an n-ary decomposition $(x_0, \ldots, x_{\ell-1})$ of x, and proving that $x = \sum_i x_i \cdot n^i$ and each x_i belongs to $[0, n)$ (which can be done efficiently when n is small). The state of the art method in this paradigm is Bulletproofs [BBB+18], which features very small proof size $O(\lambda \cdot \log \ell)$ for a security parameter λ (using binary decomposition), and also enjoys a transparent setup: the only trusted parameter it requires is an unstructured common random string, which can be easily generated by standard "nothing up my sleeve" methods (in contrast, protocols requiring a structured common string need to trust the parameter generator, which is undesirable). Due to its

great concrete efficiency and its transparent setup, Bulletproofs have become the most commonly used solution in real-world applications.

(2) Range proofs based on square decomposition [Bou00; Lip03; Gro05; CPP17], where one proves a statement of the form $x \geq 0$ by using special *integer commitment schemes* [FO97; DF02] to commit to x over $\mathbb{Z}$, and by proving the existence of four squares $x_1, \ldots, x_4$ such that $x = \sum_i x_i^2$ (such a decomposition always exist by a theorem of Lagrange, and ensures non-negativity). This generalizes to arbitrary intervals $[a, b]$ by proving non-negativity of $(x - a)(b - x)$. While avoiding n-ary decomposition is attractive, instantiating integer commitments required until recently the use of hidden order groups (such as RSA groups), whose elements are too large to be competitive with Bulletproofs for any reasonable interval size, and which require a trusted setup (to set up the RSA modulus).

The CKLR Range Proof. In a recent work [CKLR21b], Couteau *et al.* revived the square decomposition paradigm, by constructing *bounded* integer commitment schemes, which can be instantiated over cryptographic groups with hard DLOG problem. They instantiate (a variant of) the range proof of [CPP17] with this new commitment scheme, significantly reducing their size and removing the need for a structured common reference string. The CKLR scheme was shown to compare favorably with Bulletproofs: for a careful choice of parameters and underlying group, the proofs are about 15% shorter than Bulletproofs, and require an order of magnitude less group operations. Therefore, on paper, CKLR seems to offer a competitive alternative to Bulletproofs.

CKLR versus Bulletproofs. However, this cost estimation ignores several important practical aspects, and the distinction turns out to be far from clear cut in real-world instantiations. The main limitation of CKLR is that it requires exotic group sizes – typically, elliptic curves with elements of size 352 or 416 bits to achieve 128 bits of security for 32- or 64-bit ranges. While in theory, we can use curves with a wide variety of sizes, and many standard options exist, the vast majority of cryptographic applications build upon 256-bit elliptic curves, and highly optimized implementations of some of these curves are available (for example in libsecp256k1 [Wui18] or ristretto255 [VGT+19]). These libraries typically offer runtimes 10 to 20 times faster than the NIST standardized implementations of other standard curves. Hence, the use of large curves in CKLR actually negates the efficiency gains of their smaller number of group operations compared to Bulletproofs. Furthermore, several applications constrain the choice of curve; for example, the Ethereum cryptocurrency only allows the curve secp256k1.

This is not the only limitation of the CKLR range proof, compared to Bulletproofs. The latter is especially attractive when performing several range proofs at once, because it allows for very efficient batching of multiple proofs; no such batching is known for CKLR. This stems from the fact that the CKLR range proof revolves around an "extraction lemma" which was formulated and proven in the setting of a single proof, and operates on top of single-value commitments (while Bulletproofs operate on generalized Pedersen commitments, which can commit compactly to vectors of values).

Eventually, CKLR is also more restricted in its range of applications compared to Bulletproofs. This is because Bulletproofs operate with standard Pedersen commitments, while CKLR is designed on top of a new (Pedersen-based) construction of bounded integer commitments. Compared to Pedersen commitments, these new commitments have (1) only limited homomorphic properties, and (2) a relaxed notion of opening, where a malicious opener is given more freedom in what is regarded as a valid opening (this is similar in spirit to the property of standard integer commitment schemes, such as the Damgård-Fujisaki commitment [DF02]). This means that in some applications, for example when a value opened by a malicious party must be reused afterwards by an honest prover (this is the case, e.g. in some cryptocurrency applications), CKLR cannot be used as a drop-in replacement: the use of CKLR is only appropriate when the new commitment scheme can be used in the application without harming security or correctness.

Summing up, the CKLR paradigm is a promising new approach for constructing range proofs with strong performance. However, it does not currently compare favorably to Bulletproofs in practical applications, mostly due to its use of larger curves which lack competitive implementations, but also due to its lack of batching features. Furthermore, it operates on a new commitment scheme, which makes it not a priori clear what are the standard applications of range proofs where it can be safely used.

3.1.1. Our Contributions

In this work, we thoroughly revisit the CKLR paradigm. We introduce a new family of range proof schemes, which we call Sharp (for short relaxed range proofs). The name Sharp stems from a change of perspective with respect to CKLR: in CKLR, a proof is interpreted as a full-fledged range proof for values committed with a new *bounded integer commitment* which they introduce. The latter is essentially a Pedersen commitment where openings are allowed to be *rationals*, which are rounded to the nearest integer in the opening phase. We observe that one can equivalently "push the relaxation from the commitment to the range proof" and see CKLR as a *relaxed* range proof operating over standard Pedersen commitments, where *relaxed* means that the prover is only bound to a *rational* inside the target range, instead of an integer.[1] While this change of perspective does not in itself change the construction nor its security properties, it allows for a more modular treatment of the construction, and simplifies the analysis of how CKLR (or Sharp) integrates within standard application of range proofs.

Our new constructions build upon numerous optimizations, which are a combination of known techniques and entirely new approaches. The security analysis of our scheme is subtle and technically involved; it forms the core technical contribution of our work. Sharp proofs improve upon CKLR on all possible fronts: they are much shorter, more efficient, allow for a considerably more flexible choice of the underlying group (and can in particular be efficiently instantiated over 256-bit curves), and can be batched efficiently. In addition, we also demonstrate how to overcome the relaxation of soundness, obtaining schemes that operate directly with standard Pedersen commitments and effectively bind the prover to an *integer* in the range (instead of a rational) at the cost of slightly larger proofs (but still with very competitive performance).

To complement the above results, we elaborate on how Sharp can be used to improve the efficiency of some flagship applications of range proofs, such as anonymous credentials and anonymous transactions, clarifying which applications can work with bounded integer commitment schemes, and which require using a scheme with stronger features. We validate our efficiency claims with implementations and benchmarks of our main schemes. While our implementation is an unoptimized proof-of-concept implementation, our benchmarks show that it offers a ten-fold runtime improvement over a heavily optimized implementation of Bulletproofs; we expect that the efficiency gap would widen further with a more optimized implementation of Sharp. Below, we elaborate on our contributions.

[1] This is a purely conceptual change of view with respect to CKLR, where the rational opening is afterwards interpreted as an encoding of the closest integer via rounding.

3.1.1.1. Improved Range Proof Constructions.

Our new family of range proofs, Sharp, can be instantiated in a variety of settings, leading to tradeoffs between efficiency and the underlying soundness notion. We build upon the paradigm introduced in [CKLR21b] and obtain range proofs with improved efficiency and flexibility. In applications where low communication matters the most, our scheme $\mathsf{Sharp}_{\mathsf{GS}}$ provides the most competitive performance, but uses curves of sizes other than the standard 256-bit setting. For runtime-critical applications, or when the application restricts the available curve, we describe $\mathsf{Sharp}_{\mathsf{SO}}^{\mathsf{Po}}$, a scheme fully optimized to work over 256-bit groups.

At the heart of our flexibility and efficiency improvements is a modular treatment of the structure of a range proof. We split the range proof into two conceptual parts: the proof of short opening (PoSO) and the proof of decomposition (PoDec). The PoSO guarantees that extracted openings are short and the PoDec ensures that the square decomposition holds over $\mathbb{Z}_p$, where p is the order of the DLOG group. Combining both parts ensures that the committed value is a *rational* inside the given range, as the shortness allows us to argue over the integers. This decoupling allows us to develop tailored optimizations for each part, but also clarifies the exact soundness guarantees which the proof provides. We stress that one can still equivalently see Sharp as a standard range proof operating over a relaxed integer commitment scheme, using the rounding technique of CKLR: our change of perspective improves the conceptual simplicity of analyzing the use of Sharp within standard applications, but the exact guarantees remain identical to CKLR.

Optimizing the decomposition proof. We optimize the PoDec via a polynomial-based technique, similar to the lattice version of [CKLR21b] (with some tweaks that improve efficiency). Besides improving efficiency of the PoDec, this adaption enables two additional improvements: (1) The new protocol is suited for vector commitments, such as Pedersen multi-commitments (MPed). This enables more efficient batch range proofs, in the sense of performing range proofs for all N values in the vector commitment at once. (2) We introduce a group switching strategy that enables the use of different groups for the PoSO and PoDec. To our knowledge, this is the first time group switching is (efficiently) used without leveraging hidden order groups. This optimization further reduces proof length (and computation), while allowing more flexibility to instantiate the underlying groups. These changes lead to an optimized range proof: $\mathsf{Sharp}_{\mathsf{GS}}$.

Optimizing the short opening proof. We further present $\mathsf{Sharp}_{\mathsf{SO}}^{\mathsf{Po}}$, a range proof with optimized PoSO (in combination with the changes described above). The analysis of this scheme is delicate and uses several new ideas. It constitutes the main technical contribution of this work. As range and challenge space (hence soundness) introduce lower bounds on group size, repetitions are required to achieve high security levels when the group is fixed. In CKLR, such repetitions were very expensive, as much of the proof had to be repeated. To reduce their cost, we introduce a (fractional) shortness test that allows the prover to show that numerator and denominator of multiple fractions are *short* by sending a single *short* integer, per repetition. Integrating this shortness test in the range proof, a “repetition” requires only two scalars, independent of the batch size. Thus, the bulk of communication and computation of the range proof is the optimized PoDec (*without* any repetition).

We note that these optimizations also lead to significant improvements in a batch setting, where multiple range proofs must be executed at once. For example, executing $N = 8$ range proofs with 128 bits of security and 64-bit inputs communicates only 2.9 times more than executing a single range proof. We also observe that a similar batch technique is used in the context of lattice-based range proofs, in the setting where all challenges are bits. However, the possibility of using general short challenges instead of bits is precisely what allows our schemes to remain very compact, and is also what makes the analysis of our shortness test so delicate (we elaborate on this aspect in the technical overview).

Binding to integers instead of rationals. The bounded integer commitment scheme of [CKLR21b] is essentially a Pedersen commitment where malicious openers are allowed to reveal a rational instead of an integer (that is later rounded to encode an integer inside the range). Consequently $\mathsf{Sharp}_{\mathsf{GS}}$, like CKLR, provides only a relaxed notion of soundness, in that it only binds the prover to a rational in the target range. We develop several new approaches to overcome this limitation, obtaining proofs that operate with standard Pedersen commitments (where openings are required to be integers). In the interactive setting, where soundness is statistical (and a 2^{-40} statistical soundness error is a common choice), we show how our batch shortness test allows us to use challenges in $\{0, 1\}$ with much more reasonable communication overhead compared to previous approaches, which gives a competitive three-round range proof with transparent setup and full-fledged soundness. In the non-interactive setting (where soundness is computational and 128 repetitions would be too expensive), we show how to combine our schemes with a minimal use of hidden order groups, obtaining two variants: $\mathsf{Sharp}_{\mathsf{CL}}$ (using class groups to instantiate the hidden order group) and $\mathsf{Sharp}_{\mathsf{RSA}}$ (using RSA groups). These variants retain a strong efficiency, as only a *single element* of the hidden order group must be added to the proof. They achieve stronger soundness notions, namely: (1) $\mathsf{Sharp}_{\mathsf{RSA}}$ achieves standard soundness (allowing our scheme to be used as a drop-in replacement in essentially any application of range proofs, but at the cost of loosing the transparent setup), and (2) $\mathsf{Sharp}_{\mathsf{CL}}$ achieves a slightly weaker soundness where the prover is bound to a *dyadic rational*, which suffices to overcome some attacks that arise from the use of a range proof with relaxed soundness in some applications, while retaining the transparent setup.

We note that many range proofs in RSA groups have been described in the past [Bou00; Lip03; Gro05; CPP17]. Our RSA-based variant achieves considerable efficiency improvements compared to all these previous works (both communication and computation-wise), while achieving the same soundness guarantees.

Concrete efficiency estimations. We compare the communication efficiency of $\mathsf{Sharp}_{\mathsf{GS}}$, $\mathsf{Sharp}_{\mathsf{SO}}^{\mathsf{Po}}$, and $\mathsf{Sharp}_{\mathsf{RSA}}$ to the state-of-the-art in Table 3.1, and provide further tables in Appendix A.6. For performing a single range proof, $\mathsf{Sharp}_{\mathsf{GS}}$ proofs are almost 50% shorter than Bulletproofs, and about 34% shorter than the CKLR range proofs. For our computation-optimized range proofs $\mathsf{Sharp}_{\mathsf{SO}}^{\mathsf{Po}}$, these numbers are about 42% and 29% respectively. When performing a large number of range proofs, Bulletproofs become better communication-wise, because of their logarithmic cost in the batch size; nevertheless, even for a batch of $N = 8$ range proofs, our range proofs are only between 1.1 and 1.3 times larger than Bulletproofs (in concrete applications, we believe that this should be largely compensated by our strong computational improvements). Our variant in RSA groups, which achieves standard soundness, improves by a large margin compared to the previous best-known RSA-based range proof of [CPP17]: a factor 3 improvement for a single range proof, and up to a factor 14 improvement for $N = 8$ simultaneous range proofs.

We implemented our computation-optimized range proof $\mathsf{Sharp}_{\mathsf{SO}}^{\mathsf{Po}}$, using the 256-bit elliptic curve from the libsecp256k1 library [Wui18]. We stress that this is an unoptimized implementation; yet, compared to the optimized reference implementation of Bulletproofs using the same library, and running the two protocols on the same machine, we observe very significant runtime improvements. The runtime of our prover is 11 to 17 times faster than Bulletproofs' (for 32-bit and 64-bit ranges), while our verifier is two to four times faster; see Table 3.2. For a larger batch size of $N = 8$, our verifier runtime remains two to four times faster than Bulletproofs, while the gap with our prover runtimes increases slightly, ranging from 11 to 21 times faster (all while maintaining a proof size only 1.1 to 1.3 larger than that of Bulletproofs for $N = 8$). We expect these gaps to further increase with a more optimized implementation.

Table 3.1.: Theoretical proof size in Bytes for showing that some $x \in [0, B]$ of CKLR proofs [CKLR21b], Bulletproofs [BBB+18], RSA-based range proofs [CPP17] and Sharp proofs ($\mathsf{Sharp}_{\mathsf{GS}}$, $\mathsf{Sharp}_{\mathsf{SO}}^{\mathsf{Po}}$ and $\mathsf{Sharp}_{\mathsf{RSA}}$) given the security parameter λ. The groups $\mathbb{G}_{\mathsf{com}}$ and $\mathbb{G}_{\mathsf{3sq}}$ used for Sharp proofs have order p and q respectively. π denotes proof size in Bytes, N denotes the number of proofs in the batch, and $\log p, \log q$ is the bit-size of p and q.

		CKLR		BPs	RSA	$\mathsf{Sharp}_{\mathsf{GS}}$			$\mathsf{Sharp}_{\mathsf{SO}}^{\mathsf{Po}}$			$\mathsf{Sharp}_{\mathsf{RSA}}$
$(\lambda, \log B)$	N	$\log p$	π	π	π	$\log p$	$\log q$	π	$\log p$	$\log q$	π	π
128, 64	1	416	545	672	2424	333	411	360	256	256	389	793
	8	416	4360	864	19056	333	411	1070	256	256	1119	1503
	16	416	8720	928	38064	333	411	1882	256	256	1928	2315
128, 32	1	352	501	608	2404	301	347	318	256	256	335	751
	8	352	4008	800	18896	301	347	916	256	256	932	1349
	16	352	8016	864	37744	301	347	1600	256	256	1612	2033

Table 3.2.: Benchmark of our optimized range proofs compared to Bulletproofs, using the reference Bulletproofs implementation in *C* of [BBB+18], using batch sizes $N = 1$ and $N = 8$. Both implementations use the library libsecp256k1 [Wui18], and were run on a MacBook Pro with a 2.3 GHz Intel core i7 processor. All timings are in milliseconds.

		Bulletproofs		$\mathsf{Sharp}_{\mathsf{SO}}^{\mathsf{Po}}$	
$(\lambda, \log B)$	N	Prover's work	Verifier's work	Prover's work	Verifier's work
128, 64	1	20.6	2.55	1.17	0.75
	8	157	12.1	7.47	3.88
128, 32	1	10.5	1.46	0.97	0.74
	8	80.0	6.93	6.74	3.39

3.1.1.2. Security and Applications.

We analyze the guarantees of range proofs with relaxed soundness (such as CKLR and Sharp) in standard range proof applications. For this, we show which manipulations of the committed values can be allowed depending on the setting. Specifically, we discuss the arithmetical behaviour of the manipulated rationals, the impact of the chosen decomposition on soundness and show that Sharp proofs provide standard soundness when the committed values are short. Then, we use these insights to sketch how Sharp can be applied to two important applications of range proofs: anonymous credentials (AC) and anonymous transactions (AT). While relaxed soundness is sufficient in AC, range proofs with relaxed soundness do not suffice as drop-in replacement in AT (and their usage would lead to concrete attacks). Nevertheless, some (but not all) range proofs can be replaced with Sharp proofs in AT, and we sketch how Sharp proofs augmented with both a RSA and class group element improve this situation, even *without* trusted setup of the RSA modulus.

3.1.2. Technical Overview

3.1.2.1. CKLR Proofs.

Before introducing our technical improvements, we give a short overview of CKLR in the DLOG setting. Given a group $\mathbb{G}$ of order p with generators (G, H), a Pedersen commitment (Ped) to $x \in \mathbb{Z}_p$ with randomness r is given by $xG + rH$. (We use additive notation.)

CKLR opens the commitment to $x \in [0, B]$ in a zero-knowledge manner using standard Σ-protocol techniques. That is, the prover commits to random masks in $D = \mathsf{Ped.Com}(\widetilde{x}, \widetilde{r})$, where $\widetilde{x}$ and $\widetilde{r}$ are additive masks for x and r respectively. Then, sends D to the verifier who in turn sends a random challenge $\gamma \in [0, \Gamma]$. The prover responds with two linear combinations $z = \gamma x + \widetilde{x}$, $t = \gamma r + \widetilde{r}$. Finally, the verifier checks the linear combination via $D + \gamma C = \mathsf{Ped.Com}(z, t)$ and checks $z \in [0, (B\Gamma + 1)L]$, where L is the "masking overhead". We call such a "proof of opening with shortness check" a *proof of short opening (PoSO)*.

The basic observation in [CKLR21b] is that the soundness of the above protocol guarantees the extraction of a value of the form $x \equiv_p y \cdot \gamma^{-1}$, where both (y, γ) are short as well. While this does not suffice to bind the prover to a small integer, CKLR observes that $x \equiv_p y \cdot \gamma^{-1}$ uniquely defines a small rational number $u = y/\gamma \in \mathbb{Q}$ (where y, γ are short and coprime), if $2(B\Gamma + 1)\Gamma L \leq p$ holds.[2] We call $u \in \mathbb{Q}$ the *rational representative* of x and write $u = [x]_{\mathbb{Q}}$.

To show that u resides in the range $[0, B]$, CKLR decomposes $x(B - x) = \sum_{i \in [1,4]} y_i^2$ as the sum of four squares, commits to y_i in separate Ped commitments, performs a PoSO for the y_i and x, and shows that the decomposition holds over $\mathbb{Z}_p$ using the homomorphic properties of Ped. We call this part a *proof of decomposition (PoDec)*. The shortness guarantees of the PoSO imply that $u(B - u) \geq 0$ and thus $u \in [0, B]_{\mathbb{Q}}$, if $18((B\Gamma + 1)L)^2 \leq p$ holds.[3]

3.1.2.2. $\mathsf{Sharp}_{\mathsf{GS}}$: Group Switching and Batching via an Adapted PoDec.

To weaken the requirements on commitment homomorphism, we use a polynomial-based technique. That is, the prover commits to y_i in Ped commitments and performs a PoSO for each y_i, as before. To show that the four square decomposition holds, i.e. $x(B - x) = \sum_{i \in [1,4]} y_i^2$, the prover computes a polynomial f using the (short) masked witnesses $z = \gamma x + \widetilde{x}$ and $z_i = \gamma y_i + \widetilde{y}_i$ from the PoSO as follows:

$$f = z(\gamma B - z) - \sum_{i=1}^{4} z_i^2 = \alpha_2 \gamma^2 + \alpha_1 \gamma + \alpha_0.$$

A short computation shows that $\alpha_2 = 0$, i.e. the degree of f in γ is 1, iff the decomposition holds. To show that the degree of f is one, the prover commits to α_1 and α_0 in $C_* = \mathsf{Ped.Com}(\alpha_1; r_*)$ and $D_* = \mathsf{Ped.Com}(\alpha_0; \widetilde{r}_*)$ and sends C_*, D_* to the verifier. Then, the verifier sends the challenge γ and the prover replies with $t_* = \widetilde{r}_* + \gamma r_*$. Note that the verifier can recompute f from z, $\{z_i\}_{i=1}^4$ and the statement. Now, the verifier can check whether $f \equiv_q \alpha_1 \gamma + \alpha_0$ via $\mathsf{Ped.Com}(f, t_*) = D_* + \gamma C_*$. As the challenge is not known to the prover at the point of committing to the coefficients, the Schwartz–Zippel lemma guarantees that the decomposition holds over $\mathbb{Z}_q$ with overwhelming probability. Further, the

[2] CKLR interprets (y, γ, r) as a valid opening to u with respect to a modified Pedersen commitment that commits to rationals $u = y/\gamma$ as $(y \cdot \gamma^{-1})G + rH$ (or integers with rounding). Instead of relaxing the commitment, we relax the soundness guarantee of the range proof and keep working over rationals. This is more flexible and precise.

[3] For improved efficiency, CKLR and our protocols actually use a three square decomposition which can lead to problems in applications, see Section 3.6.1.2. For simplicity, we stick with the four square decomposition in the introduction.

prover reveals nothing about the values as the commitments are hiding and the openings are masked in t_*.

By construction, the polynomial-based technique allows us to use Pedersen multi-commitments (MPed), instead of separate Pedersen commitments (as in CKLR). Thus, we can perform N range proofs at once, with a constant number of group elements and a linear number of *short* integers.

The high level structure of this Σ-protocol resembles the lattice-based version of CKLR. But now, by committing to the entire decomposition y_i in a *single* Pedersen *multi*-commitment, which was not possible in the DLOG Σ-protocol of CKLR, the prover needs to communicate two integers and group elements fewer, compared to CKLR. This improves over the standard Σ-protocol for the showing the square decomposition in a group setting [CPP17; CKLR21b].

Group Switching. We highlighted in the overview above that the uniqueness of rational representatives requires (only) that $p \geq 2(B\Gamma + 1)\Gamma L$. Unfortunately, for the guarantee that the 3-square decomposition holds, this becomes $p \geq 18K^2$, where $K = (B\Gamma + 1)L$, which almost doubles the minimal possible group size. We observe that a dependency of PoSO and PoDec, which was present in CKLR, is removed with our improved Σ-protocol. Thus, we can choose groups with different modulus for the PoSO and PoDec. This gives us flexibility in group choices, and no compromise between optimal choice for commitment (typically 256-bit groups) or PoDec (typically larger groups) has to be made.

3.1.2.3. $\mathsf{Sharp}^{\mathsf{Po}}_{\mathsf{SO}}$: Cheaper Repetitions via a Novel PoSO

To clarify the requirements for our PoSO, we take a closer look at the security proof of $\mathsf{Sharp}_{\mathsf{GS}}$. The PoDec proves (among other equations) the square decomposition of N integers x_i:

$$x_i(B - x_i) \equiv_p \sum_{j=1}^{4} y_{i,j}^2 \tag{3.1.1}$$

for each committed value x_i. Security of PoDec follows from 3-special soundness, i.e. 3 related transcripts. To derive that $[x_i]_{\mathbb{Q}} \in [0, B]_{\mathbb{Q}}$, the security proof exploits a guarantee of the (simple) PoSO: Given two related transcripts $(a, \gamma, \vec{z})$ and $(a, \gamma', \vec{z}')$, we can extract $x_i \equiv_p \overline{z}_i/d$ where $\overline{z}_i = z_i' - z_i$ and $d = \gamma' - \gamma \in [-\Gamma, \Gamma]$, and likewise for $y_{i,j}$; given a third related transcript, Eq. (3.1.1) is ensured. Moreover, $\overline{z}_i \in [-K, K]$ due to verifier size checks, so $[x_i]_{\mathbb{Q}} = \frac{\overline{z}_i}{d} \in \mathbb{Q}_{K,\Gamma}$, i.e. a fraction with numerator bounded by K and denominator bounded by Γ. Thus, multiplying Eq. (3.1.1) by d^2, it is a homogeneous quadratic equation in d, B, $\overline{z}_i$, and $\overline{z}_{i,j}$, all of which bounded by K, so short. Since $18K^2 < p$, the equation holds over the integers. As a consequence, any PoSO which ensures that all extracted x_i, $y_{i,j}$ are of the form $x_i = \overline{z}_i/d$ and $y_{i,j} = \overline{z}_{i,j}/d$ is sufficient for this argument. Note that it is important that all fractions x_i, $y_{i,j}$ share the same denominator d for the above argument. Thus, we aim to replace the individual PoSOs by a “Batch-PoSO”: Given any number of x_is (where we do not distinguish between x_i and $y_{i,j}$ anymore), prove that all of them are short fractions (i.e. in $\mathbb{Q}_{K,\Gamma}$) with a shared denominator d.

A straightforward approach is the following: To check shortness of $x_1, \ldots, x_N$, check shortness of the random linear combination $S = \sum_i \gamma_i x_i$ for $\gamma_i \leftarrow [0, \Gamma]$ (where we ignore masking terms for zero-knowledge for simplicity). Intuitively, if any x_i is not short,[4] the term $\gamma_i x_i$ should ensure that S is not short with high probability. And indeed, it is not hard to see that *individually*, every x_i is of the

[4] Recall that, e.g. $1/d \in \mathbb{Z}_p$, is considered short for $d \leq \Gamma$ in our setting.

form $\overline{z}_i/d_i$ for short $\overline{z}_i$ and d_i, where $d_i \in [1, \Gamma]$. However, as we explained above, we require that the *common denominator* d of all $\overline{z}_i/d_i$ is also short. Perhaps surprisingly, this does not follow trivially.

It is clear that, by using *binary* challenges, i.e. $\Gamma = 1$, all d_i are 1, and thus, the common denominator d is 1. In fact, all $\overline{z}_i/d_i = \overline{z}_i$ are small *integers*. This simple approach is well-known and used in (lattice-based) cryptography for proving knowledge of short preimages via random subset sums. While this even ensures standard soundness, it has the huge drawback of a binary challenge space. Thus, 128 repetitions are required for knowledge error 2^{-128}, which leads to relatively large proof size, e.g. instead of a 335-byte (relaxed sound) we get a 1877-byte (standard sound) range proof from $\mathsf{Sharp}_{\mathsf{SO}}^{\mathsf{Po}}$ (for 32-bit range).

To achieve the claimed proof size, we must therefore choose a large challenge space $[0, \Gamma]$, so as to minimize repetitions. The crux of the security proof is then to ensure the common denominator d of all $\overline{z}_i/d_i$ is still short. Our core lemma (Lemma 3.3.12) asserts, that either such a short common d exists, or the false acceptance probability at most $8/\Gamma$, This result is surprisingly non-trivial to prove, and it may be of independent interest.

Relation to similar lattice-based approaches As noted before, our Batch-PoSO bears close similarities to some (approximate) batch proofs of (knowledge of) short preimages in the lattice setting. Indeed, random linear combinations for batch proofs are a standard approach and used in the lattices setting, e.g. with binary challenges in [BBC+18]. It is also used with larger challenges spaces to prove "fractional openings" of commitments, resulting in relaxed soundness somewhat similar to our setting, e.g. in [BKLP15; BDL+18]. Namely, by multiplying with the (small) denominator, an extracted solution grows in size, but if parameters are chosen accordingly, the lattice problem still remains hard even for such larger solutions. Moreover, in special settings, e.g. ring-lattices, special challenge sets $\mathcal{C}$ where even $(\gamma' - \gamma)^{-1}$ is small for all $\gamma, \gamma' \in \mathcal{C}$ are used [AL21].

However, a crucial difference between our setting and the lattice-setting is that, in all the lattice-based works we are aware of, the challenge space for proving (approximate or relaxed) shortness is small and a large number of repetitions are required. Moreover, in these works, there is no requirement for a short *common* denominator d, instead, it suffices that *individually* each d_i is small, which is straightforward to show (but insufficient in our case). Since we embrace relaxed soundness and aim to maximize the challenge space, our approach exhibits such a requirement. Hence, to prove security, we require an entirely new analysis for the random linear combination test. Our current proof seems quite different from (advanced) lattice-based techniques, but it is an interesting question if and how such techniques are applicable to strengthen the lemma or simplify its proof.

Lastly, we note that lattice-based proof systems have vastly improved; even exact (range) proofs are now quite small, e.g. [LNS20; LNP22], though still an order of magnitude larger than group-based proofs, e.g. [LNP22] notes that a proof of opening alone needs 8 kB. We leave it as an interesting question, whether lattice-based range proofs could benefit from square-decompositions or our techniques as well.

3.1.2.4. $\mathsf{Sharp}_{\mathsf{HO}}$: Augmenting Sharp with Hidden Order Groups.

By using groups of hidden order, we can achieve improved soundness guarantees. On a high level, we add a single MPed commitment C' in a hidden order group to Sharp to restrict the possible commitment openings to "special" rationals. In contrast, all other range proofs in hidden order groups perform the *entire* range proof in the hidden order group [Bou00; Lip03; Gro05; CPP17; CKLR21b]. As these groups are larger than standard DLOG groups, our approach heavily improves efficiency.

Our proof of opening for the additional commitment only requires one additional short integer (for proving knowledge of the randomness of C'), as we use a *synthesized* challenge γ' and response z_i' (computed from the actual challenges and responses) to avoid further repetitions (even if the underlying range proof is repeated). In more detail, when the PoSO is repeated R times with challenges $\{\gamma_k\}_{k=1}^R$, the prover and verifier set $\gamma' = \sum_{k=1}^R \gamma_k (\Gamma + 1)^{k-1}$ and similarly for z_i'. So for completing the proof, only the masked commitment randomness t_x' is sent additionally. When instantiating this augmentation with suitable class groups, the committed x_is are restricted to be dyadic rationals, i.e. of the form $m/2^\ell$. With RSA groups, the x_i must be integers, hence the proof is standard sound.

3.1.3. Structure of this Chapter

In Section 3.2, we introduce additional basic definitions and notations. Section 3.3 contains our notion of shortness testing and the soundness result for our random linear shortness test. In Section 3.4, we present our improved range proof which allows group switching. Then, in Section 3.5, we show how to combine these improvements with our batch shortness test. We discuss properties of relaxed soundness how to augment our soundness by using hidden order groups in Section 3.6. Lastly, we discuss possible applications of our results in Section 3.7.

3.2. Preliminaries

In this section, we introduce the additional preliminaries required for this chapter. See Chapter 2 for the global preliminaries.

3.2.1. Notation and Basic Functions

We write $[a, b]$ for an interval $[a, b]$ in $\mathbb{Z}$, and we write $[a, b]_R$ for an interval in another space R, e.g. $\mathbb{Q}, \mathbb{R}, \mathbb{Z}_p$. We use Minkowski sum notation for sets, i.e. $A + B := \{a + b \mid a \in A, b \in B\}$ and write $A + b := A + \{b\}$ for offsets. We denote by $|x|$ the absolute value of $x \in \mathbb{R}$. Let p be an (odd) (prime) number. Let $\mathbb{Z}_p = \mathbb{Z}/p\mathbb{Z}$ be the integers modulo p, with representatives either $\mathbb{Z}_p = [0, p-1]$ or $\mathbb{Z}_p = [\lceil -\frac{p-1}{2} \rceil, \lceil \frac{p-1}{2} \rceil]$. Generally, we write $\equiv_p$ for equality $\bmod p$ and $\in_{\mathbb{Z}_p}$ for set membership modulo p, i.e. $x \in_{\mathbb{Z}_p} S$ iff $\exists s \in S\colon x \equiv_p s$. For $x \in \mathbb{Z}_p$, let $|x| = \min\{|k| \mid k \in \mathbb{Z}, k \equiv_p x\} \le p/2$. Recall that $d(x, y) = |y - x|$ for $x, y \in \mathbb{Z}_p$ defines a metric on $\mathbb{Z}_p$.

We define the "prime number analogue" of the factorial.

Definition 3.2.1 (Primorial). We write $\mathsf{priml}(k)$ for the product of the first k primes, i.e. $\mathsf{priml}(k) := \prod_{i=1}^k p_i$ where p_i is the i-th prime number.[5] We write $\mathsf{primlmin}(n)$ for $\min\{k \mid \mathsf{priml}(k) \ge n\}$, i.e. the smallest k such that $\mathsf{priml}(k) \ge n$.

[5] The usual definition of *primorial* is $n\# = \prod_{p_i \le n} p_i$, where p_i is the i-th prime. That is, $n\#$ is the product of all primes p_i up to n. Thus, $\mathsf{priml}(k) = p_k\#$.

3.2.2. DLSE and SEI Assumptions in Cryptographic Groups

Instead of the usual DLOG assumption, we consider a version for arbitrary ranges of secret exponents. This allows us to consider exponents which are (much) smaller than the group size, which is important to improve efficiency in large groups, e.g. in RSA groups. Moreover, we consider a corresponding indistinguishability assumption, the *short exponent indistinguishability* assumption, which asserts that it is hard to distinguish random group elements with short exponents from random group elements.

Definition 3.2.2 (S-Bounded DLSE and SEI). Consider a group $\mathbb{G}$. The S-bounded **discrete logarithm with short exponents** (DLSE) assumption holds if for every PPT $\mathcal{A}$ there is a negligible function negl such that

$$\Pr\left[G \xleftarrow{\$} \mathbb{G}; z \xleftarrow{\$} [0,S],\ z' \leftarrow \mathcal{A}(G, zG)\colon\ z = z'\right] \le \mathsf{negl}(\lambda)$$

This probability defines the advantage $\mathsf{Adv}^{\mathrm{dlse}}_{\mathcal{A}}$ of $\mathcal{A}$ against DLSE.

The S-bounded **short exponent indistinguishability** (SEI) assumption holds if for all PPT $\mathcal{A}$ there is a negligible negl function such that

$$\begin{aligned}&\Pr\left[G \xleftarrow{\$} \mathbb{G}; z \xleftarrow{\$} [0,S]\colon \mathcal{A}(G, zG) = 1\right]\\ -&\Pr\left[G \xleftarrow{\$} \mathbb{G}; z \xleftarrow{\$} \mathbb{Z}_{\mathsf{ord}(G)}\colon \mathcal{A}(G, zG) = 1\right]\\ \le&\mathsf{negl}(\lambda)\end{aligned}$$

This probability defines the advantage $\mathsf{Adv}^{\mathrm{sei}}_{\mathcal{A}}(\lambda)$ of $\mathcal{A}$ against SEI.

For groups of known order p, SEI holds unconditionally for $S = p - 1$. More generally, an unbounded adversary against SEI for $S = LU_{\mathrm{up}}$ has advantage at most $1/L$ in groups of unknown order (Remark A.1.1), but relying on the SEI assumption for $S > U_{\mathrm{up}}$ is of little interest.

Note that SEI is a (long-standing) highly plausible assumption. Further, the DLSE and SEI assumption are known to be essentially equivalent in groups of known prime order with random generators [KK04], but a security loss is incurred in the reduction.

3.2.2.1. Pedersen Commitment

We consider Pedersen multi-commitments (MPed), a generalization of the Pedersen commitment scheme [Ped92], with short openings over a prime or hidden order group $\mathbb{G}$. Let $N, S \in \mathbb{N}$ and $U_{\mathrm{lo}} \le |\mathbb{G}| \le U_{\mathrm{up}}$. Setup samples $G_i \xleftarrow{\$} \mathbb{G}$ for $i \in [0, N]$ and outputs commitment key $ck = (\{G_i\}_{i\in[0,N]})$. Given a message vector $\{x_i\}_{i\in[1,N]}$, Com samples $r \xleftarrow{\$} [0, S]$, sets $C = rG_0 + \sum_{i\in[1,N]} x_i G_i$, and outputs the pair (C, r). Given commitment C, message $\{x_i\}_{i\in[1,N]}$ and opening r, $\mathsf{VfyOpen}$ outputs 1 iff $C = rG_0 + \sum_{i\in[1,N]} x_i G_i$ and x_i is in the right message space for all i. That is, if $\mathbb{G}$ has prime order p, then $x_i \in \mathbb{Z}_p$, or else $x_i \in \mathbb{Z}$ unless stated otherwise. We write Ped for the Pedersen commitment scheme, i.e. MPed for $N = 1$. The scheme MPed is hiding under the SI and SEI assumptions and binding under the DLOG assumption. The strength of the hiding property scales with hiding parameter S.[6]

[6] If $G_i \xleftarrow{\$} \langle G_0 \rangle$ and S is large enough, then MPed is statistically hiding. Under the SI assumption, instead using $G_i \xleftarrow{\$} \mathbb{G}$ remains (computationally) hiding. Usually, sampling $G_i \xleftarrow{\$} \mathbb{G}$ can be transparent (trapdoor-free), but $G_i \xleftarrow{\$} \langle G_0 \rangle$ not necessarily.

3.2.3. Rational Representatives

Using $\mathbb{Z}$-valued representatives for $\mathbb{Z}/p\mathbb{Z}$ is a natural choice, obtained from the homomorphism $\mathbb{Z} \to \mathbb{Z}_p$, $x \mapsto x \bmod p$. Another choice is induced by the ring $\mathbb{Z}_{(p)} = \{\frac{n}{d} \mid n \in \mathbb{Z}, d \in \mathbb{N}, p \nmid d\} \subseteq \mathbb{Q}$, and the homomorphism $\frac{n}{d} \mapsto n \cdot (d^{-1} \bmod p) \bmod p$. We call such representatives *rational*. Strictly speaking, a set of representatives $R \subseteq \mathbb{Z}_{(p)}$ should have a *unique* representative for each element in $\mathbb{Z}_p$. We work with smaller sets, which do not have representatives for all of $\mathbb{Z}_p$, but existing representatives are unique. The lack of surjectivity will be of no concern since, by construction, elements of interest will always come with an admissible representative.

Definition 3.2.3. Let $\mathbb{Q}_{N,D} \subseteq \mathbb{Q}$ be the rationals whose numerator is bounded by N and denominator bounded by D, that is

$$\mathbb{Q}_{N,D} = \{\frac{n}{d} \in \mathbb{Q} \mid |n| \leq N, |d| \leq D\} \subseteq \mathbb{Q}.$$

The value x is represented by $\frac{n}{d}$ if $x \equiv_p nd^{-1}$ (where d^{-1} is computed modulo p).

Note that we interpret $\frac{n}{d}$ as a fraction; the tuple (n, d) is not unique. It becomes unique if $\frac{n}{d}$ is reduced and $d \geq 1$.

Lemma 3.2.4 (Criterion for Unique Representative in $\mathbb{Q}_{N,D}$)**.** *Let N, D so that $N \cdot D < p/2$. Then for any $x \in \mathbb{Z}_p$, if there is a representative in $\mathbb{Q}_{N,D}$ of x, i.e. some $\frac{n}{d}$ so that $nd^{-1} \equiv_p x$, then $\frac{n}{d}$ is unique (as a fraction).*

Proof. Suppose $x \equiv n_i d_i^{-1} \ (p)$ for $i = 1, 2$. Then $n_1 d_2 \equiv n_2 d_1 \ (p)$. Since $N \cdot D \leq p/2$ and $\frac{n_i}{d_i} \in \mathbb{Q}_{N,D}$, we find that $n_1 d_2 = n_2 d_1$ over $\mathbb{Z}$. (No wrap-around.) Thus, $\frac{n_1}{d_1} = \frac{n_2}{d_2}$ as fractions, and the claim follows. □

From now on, we always assume that $N \cdot D < p/2$ whenever we use $\mathbb{Q}$-representatives.

Remark 3.2.5. Let $a \in \mathbb{Z}_p$ and $ND < p/2$. We define $[a]_{\mathbb{Q}} \in \mathbb{Q}_{N,D}$ as the unique irreducible representatives $\frac{n}{d}$ of a, assuming it exists. (We assume that some maximal bounds N, D are implicitly fixed in the context.) We note that $[a]_{\mathbb{Q}}$ can be efficiently computed (if it exists), see [FSW03].

3.2.4. Masking Scheme

We use "additive masking" to hide information with random noise. For readability, we use an abstraction of this technique formalized below, in a way similar to [ACK21]. A **masking scheme** is a tuple $(\mathsf{R}, \mathsf{mask}, V)$ of efficiently samplable distribution R and a masking algorithm mask for values in range $[0, V]$.

- $r \xleftarrow{\$} \mathsf{R}$ is an integer $r \in [0, (V+1)L]$, i.e. $\mathrm{supp}(\mathsf{R}) \subseteq [0, (V+1)L]$. We call r the *mask* and $L \geq 1$ the **masking overhead**.
- $\mathsf{mask}(v, r)$ takes as input an integer $v \in [0, V]$ and a mask r and outputs $v + r$ or $\perp$. For simplicity, we require $\mathsf{mask}(v, r) = \perp$ if $v + r \notin [0, (V+1)L]$.
- p denotes an upper bound on the **abort probability**, i.e. $\sup_{v \in [0,V]} \Pr[\mathsf{mask}(v, r) = \perp \mid r \xleftarrow{\$} \mathsf{R}] \leq \mathsf{p}$.
- Let M_v denote the distribution defined via: Sample $r \xleftarrow{\$} \mathsf{R}$, then return $\mathsf{mask}(v, r)$. Then $\varepsilon_{\mathsf{mask}} = \sup_{v,w \in [0,V]} \Delta(M_v, M_w)$ is called the **masking error**.

The range V is sometimes left implicit. Intuitively, $z = \mathsf{mask}(v, r)$ reveals almost nothing about v, since the random mask r ensures that z is distributed (almost) independently from v. The masking error quantifies this intuition.

Rejection Sampling. (Uniform) Rejection sampling is usually described for values in intervals $[-V, V]$, i.e. symmetric around 0. We use $[0, V]$ instead, and adapt mask accordingly. Namely, for given masking overhead L:

- The distribution R is the uniform distribution $U_{[0,(V+1)L]}$.
- $\mathsf{mask}(v, r)$ outputs $v + r$ if $v + r \in [V, (V + 1)L]$, else $\perp$.
- The abort probability is $\mathrm{p} = \frac{V+1}{(V+1)L+1} \leq \frac{1}{L}$.[7]
- The masking error is 0.[8]

Drowning in noise. In the above, set $L = 2^{\lambda}$. Then abort probability is $2^{-\lambda}$. This is convenient to use if "size" of r does not matter much.

No aborts. We also use masking schemes to save communication. In these cases, once R grows beyond $\mathbb{Z}_p$, i.e. $\mathbb{Z}_p = [0, p-1] \subseteq \mathrm{R}$, we assume that $\mathrm{R} = \mathbb{Z}_p$ and $\mathsf{mask}(v, r) = v + r \mod p$ (without abort). We will be explicit about such potential optimizations.

3.3. Shortness Testing mod p

In this section, we present a result that allows us to test shortness of many fractions at once. We will apply this result later to efficiently test shortness of committed values in our range proofs (see Section 3.5). Indeed, it is the basis for constructing a range proof which communicates a *single* integer per repetition. For readability, we only present proof sketches and sometimes simplified claims. The full claims and proofs can proofs are in Appendix A.4. First, we define a notion of "shortness test" which is tailored to our application.

Definition 3.3.1 (Fractional Shortness Test). A *(fractional) shortness test* is an algorithm T which takes as input $\vec{x} \in \mathbb{Z}_p^N$ (where T is implicitly parameterized by p and N) and outputs $T(\vec{x}) \in \{0, 1\}$. Let $K, D \in \mathbb{N}$ with $KD < p/2$. A vector $\vec{x} \in \mathbb{Z}_p^N$ is *uniformly* (K, D)*-short*, if $\exists d \in [1, D]\colon d\vec{x} \in [-K, K]^N_{\mathbb{Z}_p}$. Let $\phi_{K,D}(\vec{x}) \in \{0, 1\}$ be the predicate which is 1 if $\vec{x}$ is uniformly (K, D)-short. We say that T is a *fractionally* (K, D)*-sound* shortness test with error δ_{snd}, if

$$\forall \vec{x} \in \mathbb{Z}_p^N\colon \quad \Pr\big[T(\vec{x}) = 1 \implies \phi_{K,D}(\vec{x}) = 1\big] \geq 1 - \delta_{\mathrm{snd}} \tag{3.3.1}$$

or, equivalently,

$$\forall \vec{x} \in \mathbb{Z}_p^N : \phi_{K,D}(\vec{x}) = 0 \implies \Pr[T(\vec{x}) = 1] \leq \delta_{\mathrm{snd}}. \tag{3.3.2}$$

[7] For any $v \in [0, V]$, there are $V + 1$ "bad" r (out of $(V + 1)L + 1$ choices for r).

[8] The abort probability is independent of v. Conditioned on no abort, the distribution is uniform over $[V, (V + 1)L]$.

The crucial point in fractional (K, D)-soundness is that a vector is rejected with high probability if there is no *single* denominator of size at most D such that $d \cdot \vec{x} \in [-K, K]^N_{\mathbb{Z}_p}$, i.e. $\|d \cdot \vec{x}\|_\infty \leq K$. A weaker definition might only require $x_i \in \mathbb{Q}_{K,D}$ for all i, but this is not enough for our applications. Note that we do not define what correctness of a fractional shortness test is; it will be evident in applications and concrete requirements may vary.

Definition 3.3.2 (RAST). We define the *random affine shortness test* $\mathsf{RAST}_{N,\mathcal{D},K,\mu}$ for shortness over $\mathbb{Z}_p$ with *dimension* or *batch-size* N, *test distribution* $\mathcal{D}_N$ *range bound* K, and *offset* μ as follows: To test $\vec{x} \in \mathbb{Z}_p^N$, pick $\vec{\gamma} \xleftarrow{\$} \mathcal{D}_N$, and output 1 if $\mu + \sum_{i=1}^N x_i \gamma_i \in [0, K]_{\mathbb{Z}_p}$, else output 0.

The following theorem assures fractional soundness of the RAST. The proof is based on the core lemma, Lemma 3.3.12, whose proof is technical and lengthy; it is the subject Appendix A.4.

Theorem 3.3.3. *Let* RAST *be the random affine shortness test with uniform distribution* $\mathcal{D}$ *over* $[0, D]^N$*, dimension* N*, range bound* K*, and any offset* $\mu \in \mathbb{Z}_p$*. Let* $K' = (1 + 2\beta)K$ *where* $\beta = \min(N, \mathsf{primlmin}(D + 1))$ *and suppose that* $2D(K' + DK + 2) < p$*. Then* RAST *is fractionally* (K', D)*-sound with error* $8/(D + 1)$*,*

Remark 3.3.4 (Compressing the challenge). The verifier's challenge in RAST is relatively large, but it can be compressed. A direct reduction shows that replacing the challenge $\vec{\gamma} \xleftarrow{\$} [0, D]^N$ by $\vec{\gamma} = \mathsf{PRG}(s)$ for $s \xleftarrow{\$} \{0, 1\}^\lambda$, where PRG is a pseudo-random generator, ensures that the soundness error increases only by a negligible amount (assuming PRG is secure against *non-uniform* adversaries). And now, the verifier could send s instead, as a compressed version of $\vec{\gamma} = \mathsf{PRG}(s)$. As the security of RAST is a combinatorial property, it is an interesting question to find small (structured) challenge spaces which are unconditionally secure.

3.3.1. Modulo Arithmetic

In this section, we work with representatives $\mathbb{Z}_p = [0, p - 1]$ instead of representatives which are symmetric around 0. Mostly, because we want to use Remark 3.3.5. However, our results are phrased in a way which is independent of representatives, so they hold for any choice of representatives for $\mathbb{Z}_p$.

First, recall that a (rational) number x splits into an integer part $\lfloor x \rfloor$ and a decimal part $x - \lfloor x \rfloor$, often denoted $\mathrm{frac}(x)$.

Remark 3.3.5. For $m \in \mathbb{Z}$, $d \in \mathbb{N}$, we make much use of following simple but important equality:

$$\frac{m}{d} = \left\lfloor \frac{m}{d} \right\rfloor + \frac{m \bmod d}{d} = \left\lceil \frac{m}{d} \right\rceil - \frac{m \bmod d}{d}. \tag{3.3.3}$$

This equality holds for representatives $[0, d - 1]$ of $\mathbb{Z}_d$ for "$x \bmod p$". For modulo operations symmetric around 0, "flooring"/"ceiling" would become "rounding".

Remark 3.3.6 (Inequalities for floor and ceil). Let $x, y \in \mathbb{R}$. The we have $\lfloor x \rfloor \leq x \leq \lceil x \rceil$ and

$$\lfloor x \rfloor + \lfloor y \rfloor \leq \lfloor x + y \rfloor \leq x + y \leq \lceil x + y \rceil \leq \lceil x \rceil + \lceil y \rceil \tag{3.3.4}$$

Lemma 3.3.7 (Regular Spacing of $\mathbb{S}_d$). *Suppose* $1 < d < p$ *and* $\gcd(d,p) = 1$ *and consider the set*

$$\mathbb{S}_d \equiv_p \{\frac{i}{d} \bmod p \mid i \in [0,\dots,d-1]\} \subseteq \mathbb{Z}_p. \tag{3.3.5}$$

Then $\mathbb{S}_d = \{\lceil ip/d \rceil \mid i \in [0,\dots,d-1]\}$ *and the minimal distance* $\delta = \min_{x \neq y \in \mathbb{S}_d} |x-y|$ *satisfies* $\delta = \lfloor \frac{p}{d} \rfloor$.

When interpreting $\mathbb{Z}_p$ and the set S on the unit circle as regularly spaced points, it is visually clear that the claim should hold. See Fig. 3.1 for this. Note, $d/d \equiv_p 1$, that is, it is an angle of $2\pi/p$ away from 0, so the spacing of $\mathbb{S}_d$ is not perfectly regular. Indeed, as shown in Fig. 3.1, the points $i/d \in \mathbb{Z}_p$ are not in sequential order, but permuted by a unit modulo d, so the visual heuristics can be somewhat misleading. With Lemma 3.3.7 at hand, we can easily derive some simple consequences.

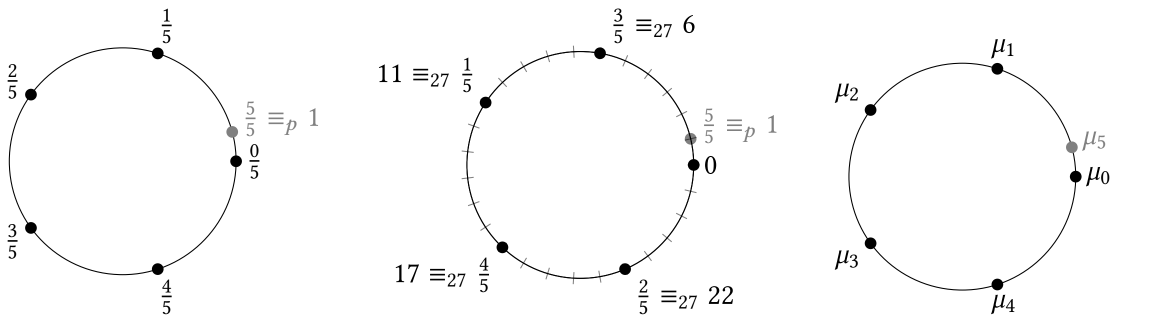


Figure 3.1.: *Visual heuristics for Lemma 3.3.7.* The left figure is the naive intuition. The middle figure is the visualization of for $p = 27$. The right figure denotes $\mathbb{S}_d = \{\mu_0,\dots,\mu_4\}$ where $\mu_i \equiv_p \lceil \frac{ip}{d} \rceil$. In the example, $\mu_1 \equiv_{27} 6$, $\mu_2 \equiv_{27} 11$, $\mu_3 \equiv_{27} 17$, $\mu_4 \equiv_{27} 22$.

Lemma 3.3.8. *Suppose* $d \in \mathbb{N}$ *and* $\gcd(d,p) = 1$ *and* $u \xleftarrow{\$} [0,\dots,d-1]$. *Let* $\mu, K \in \mathbb{N}$ *be arbitrary. Then for* $1 < d < p$ *we have*

$$\Pr\left[\frac{u}{d} \in_{\mathbb{Z}_p} [0,K]_{\mathbb{Z}_p} + \mu\right] \leq \frac{1}{d}\left\lceil \frac{K+1}{\lfloor \frac{p}{d} \rfloor} \right\rceil \tag{3.3.6}$$

and for $d > p$, *we have*

$$\Pr\left[\frac{u}{d} \in_{\mathbb{Z}_p} [0,K]_{\mathbb{Z}_p} + \mu\right] \leq 2\frac{K+1}{p} \tag{3.3.7}$$

where the probability is over u. *Combining the conditions gives*

$$\Pr\left[\frac{u}{d} \in_{\mathbb{Z}_p} [0,K]_{\mathbb{Z}_p} + \mu\right] \leq \frac{1}{d} + 2\frac{K+1}{p} \tag{3.3.8}$$

Note that in Lemma 3.3.8, we consider membership intervals $[0,K] + \mu$, i.e. arbitrary (shifted) intervals, not just $[0,K]$, because such intervals appear naturally in our setting.

While Lemma 3.3.8 was a good warm-up, it does not cover all of our needs. On the one hand, we need to deal with more general a/b (instead of just $1/d$) and $u \in [0,D]$ (instead if $u \in [0,d-1]$). On the other hand, we need to deal with unions of disjoint intervals, namely $[0,K] + \mathbb{S}_d + \mu$. Thus, we take a closer look at the specific case of interest. It is sketched in Fig. 3.2.

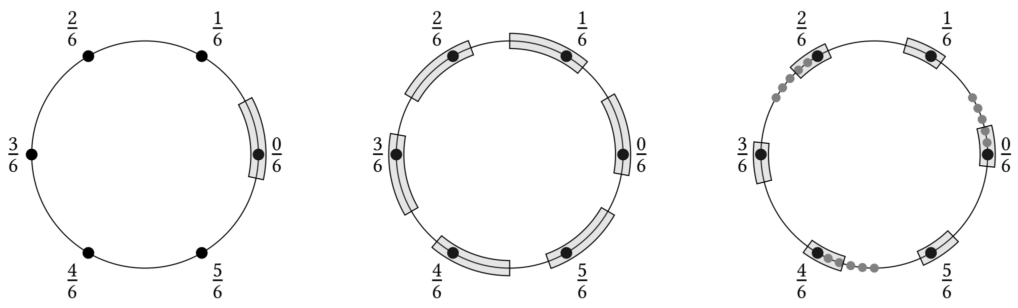


Figure 3.2.: *Visual heuristics for Lemmas 3.3.8 and 3.3.9.* The left figure is the intuition for Lemma 3.3.8. The middle figure shows $[0, K]_{\mathbb{Z}_p} + \mathbb{S}_6 + \mu$. The right figure shows $[0, K]_{\mathbb{Z}_p} + \mathbb{S}_6 + \mu$ and how points of the form $ua/3$ are distributed. Visibly, until the gray points first escape the intervals, they all lie within; after they escape, they never re-enter an interval. Hence at most $3 \cdot K/a$ points lie within $[0, K]_{\mathbb{Z}_p} + \mathbb{S}_6 + \mu$.

Lemma 3.3.9. *Let $p, d, a, b, \mu, D, K \in \mathbb{N}$ and suppose $u \xleftarrow{\$} [0, D]$ is a uniform random variable. Let $\mathbb{S}_d \equiv_p \{i/d \mid i \in \mathbb{Z}_d\} \subseteq \mathbb{Z}_p$ as usual, and likewise $\mathbb{S}_b$. Suppose that* $\gcd(d, p) = 1$, *and $b \mid d$, and that*

$$b(K + 1) + Da < \left\lfloor \frac{p}{d/b} \right\rfloor. \tag{3.3.9}$$

Then we have

$$\sum_{s \in \mathbb{S}_d} \Pr\left[u\frac{a}{b} \in_{\mathbb{Z}_p} [0, K]_{\mathbb{Z}_p} + \mu + s\right] \leq \left\lceil \frac{b(K + 1)}{a} \right\rceil \frac{1}{D + 1}. \tag{3.3.10}$$

Again, the claim of Lemma 3.3.9 is visually clear and sketched in Fig. 3.2. (Our lemma is not as tight as the picture suggests, but good enough for our purposes.)

Remark 3.3.10. A useful property of $\mathbb{S}_d$ is that $\mathbb{S}_d = 1 - \mathbb{S}_d$ (because $\frac{p-i}{d} \equiv_p 1 - \frac{i}{d}$), hence, Lemma 3.3.9 also applies to $[0, K] + \mu - \mathbb{S}_d \equiv_p [0, K] + (\mu - 1) + \mathbb{S}_d$. Moreover, Lemma 3.3.9 applies to negative a as well, where $|a|$ is used in all bounds and estimates. This follows since multiplying the expression in the probability by (-1) leads to positive n which must lie in $[-K, 0]_{\mathbb{Z}_p} - \mu - \mathbb{S}_d$ which can be rewritten as $[0, K]_{\mathbb{Z}_p} + \mu' + \mathbb{S}_d$ for suitable μ'.

Towards analyzing random linear combinations with the help of Lemmas 3.3.8 and 3.3.9, we introduce another lemma.

Lemma 3.3.11 (Simplified Lemma A.4.2)**.** *Suppose $1 \neq d \in \mathbb{N}$ and let u_i be random variables in $\mathbb{Z}_d = [0, \dots, d - 1]$ for $i = 1, \dots, N$. Fix some arbitrary $a_i \in [0, d - 1]$ with $d =$ $\mathrm{lcm}(a_1, \dots, a_N)$. Then there exist $q_1, \dots, q_N \in \mathbb{N}$, which are pairwise coprime, $q_i \mid$ $\mathrm{ord}_{\mathbb{Z}_d}(a_i)$, and $\prod_{i=1}^N q_i = d$. Let $Z = \prod_{i=1}^N \mathbb{Z}_{q_i} \hookrightarrow \mathbb{Z}_d^N$ (where the injections $\mathbb{Z}_{q_i} \hookrightarrow \mathbb{Z}_d$ of the Chinese remainder theorem is used component-wise). Then $\sum_{i=1}^N u_i \cdot a_i \mod d$ is uniformly distributed in $\mathbb{Z}_d$ for $(u_1, \dots, u_N) \xleftarrow{\$} Z$.*

Clearly, in Lemma 3.3.11, $\sum_{i=1}^N u_i a_i$ is uniformly distributed if $u_i \xleftarrow{\$} \mathbb{Z}_d$ for all i. The key point of Lemma 3.3.11 is, that the sum is uniformly distributed even if the u_i are drawn from the possibly much smaller space Z. This helps in our analysis of the core lemma.

3.3.2. Shortness Failure of Random Linear Combinations

Now, we turn to the core lemma, Lemma 3.3.12. It should be viewed as a non-trivial generalization of Lemmas 3.3.8 and 3.3.9 with certain requirements and restrictions. Implicit in Lemma 3.3.12 is the RAST from Theorem 3.3.3. That is, we consider the probability of "bad" challenges, which for a given choice of x_i's of the form $\frac{m_i}{d_i}$ lead to falsely accepting $\sum_i \gamma_i x_i$ as short, even though some x_i exceed the allowed bounds.

Lemma 3.3.12 (Core Lemma)**.** *Let $D, M \in \mathbb{N}$ and suppose $2DM < p$. Let $x_i = \frac{m_i}{d_i}$ where $d_i \in [1, D]$ and $m_i \in [-M, M]$ for $i = 1, \dots, N$. Let $\gamma_i \xleftarrow{\$} [0, D]$. Define*

$$S = \sum_{i=1}^{N} \gamma_i \cdot \frac{m_i}{d_i} \bmod p \tag{3.3.11}$$

Let $I \subseteq [1, N]$ denote the set of indices which minimizes $d := \operatorname{lcm}(\{d_i \mid i \in I\})$ under the constraint that $d > D$, or $I = [1, N]$ if $\operatorname{lcm}(d_1, \dots, d_N) \leq D$. Let $K \in \mathbb{N}$, let $\beta = \min(|I|, \mathsf{primlmin}(D + 1))$, and let $K' := K + 2\beta M$. Then, for arbitrary $\mu \in \mathbb{Z}_p$, we have

$$\Pr\big[S \in [0, K]_{\mathbb{Z}_p} + \mu\big] \leq 4 \cdot \begin{cases} \frac{1}{d} & \text{if } d(K' + 1) < p \\ \frac{1}{d} + 2\frac{K'+1}{p} & \text{always} \end{cases} \tag{3.3.12}$$

Now, suppose additionally that $d \leq D$ and $D(K' + DM + 2) < p$. If $\frac{d}{d_i}|m_i| > K'$ for some $i \in [1, N]$, then

$$\Pr\big[S \in [0, K]_{\mathbb{Z}_p} + \mu\big] \leq \frac{8}{D + 1}. \tag{3.3.13}$$

This time, we have no "visual heuristic". However, though the detailed proof of Lemma 3.3.12 is rather technical, its basic idea is relatively simple: First, simplify the situation by reducing to the case of $I = \{1, \dots, N\}$ and imposing certain minimality properties on I and d. Then, rewrite the sum S with lowest common denominator d. That is, let $S' = \sum_{i=1}^{N} \gamma_i \cdot \frac{dm_i}{d_i} \in \mathbb{Z}$, and then split S'/d (resp. S) into decimal and integer parts:

$$S \equiv_p \frac{1}{d} S' = \frac{S' \bmod d}{d} + \left\lfloor \frac{S'}{d} \right\rfloor.$$

The idea is to exploit this to analyze the two summands separately, after making them *almost* stochastically independent. (But this works only to some extent, namely, small garbage terms will appear.) For this, we change the challenge distribution. For γ_i, we could change $U_{[0,D]}$ to $U_{[0,d_i\lceil\frac{C+1}{d_i}\rceil]}$, which allows us to write $\gamma_i' = u_i + d_i v_i$ with $u_i \xleftarrow{\$} [0, d_i - 1]$ and $v_i \xleftarrow{\$} [0, \lceil\frac{C+1}{d_i}\rceil]$. Then

$$S'/d = \sum_{i=1}^{N} \gamma_i' \frac{m_i}{d_i} = \sum_{i=1}^{N} u_i \frac{m_i}{d_i} + \sum_{i=1}^{N} v_i m_i.$$

On the right hand side, the second sum is an integer sum, and relatively easy to control. The first sum has the same form as S', but the u_i are uniformly from $\mathbb{Z}_{d_i}$ now, which is simpler to analyze. However, our analysis makes use of Lemma A.4.2 to get a tighter result. Lemma A.4.2 suggests $\gamma_i' \sim U_{[0,q_i\lceil\frac{C+1}{q_i}\rceil]}$ (for suitable q_i), and we write

$$S \equiv_p \frac{1}{d} S_u + S_v \quad \text{where} \quad S_u = \sum_i u_i \frac{m_i d}{d_i} \quad \text{and} \quad S_v = \sum_i v_i q_i \frac{m_i}{d_i}.$$

The central requirements of this change in distribution are that it is close (in terms of $\rho_{\mathsf{sup}}(\vec{\gamma}/\vec{\gamma}'))$, that $S_u \mod d$ is now uniformly distributed, and that S_u and S_v are stochastically independent. Indeed, a "loss factor" of 4 compared to Lemma 3.3.8 comes precisely from $\rho_{\mathsf{sup}}(\vec{\gamma}'/\vec{\gamma})$. Moreover, when rewriting $\frac{1}{d}S_u = \frac{S_u \bmod d}{d} + \lfloor \frac{S_u}{d} \rfloor$, we can get rid of the garbage term $\lfloor \frac{S_u}{d} \rfloor = \sum_i u_i m_i \frac{q_i}{d_i}$ by increasing the interval from $[0, K]$ to $[0, K + 2\beta M]$ (since the garbage term lies in $[-\beta M, \beta M]$). After these changes, we have simplified to

$$\Pr\left[S \in_{\mathbb{Z}_p} [0,K]_{\mathbb{Z}_p} + \mu\right] \leq \rho_{\mathsf{sup}}(\vec{\gamma}/\vec{\gamma}') \cdot \Pr\left[\frac{S_u \mod d}{d} + S_v \in_{\mathbb{Z}_p} [0, K + 2\beta M]_{\mathbb{Z}_p} + \mu'\right]$$

where S_u and S_v are independent and $(S_u \mod d)$ is uniform in $\mathbb{Z}_d$. Now, Lemma 3.3.12 follows effectively from Lemma 3.3.8 (which yields Eq. (3.3.12)) and Lemma 3.3.9 (which yields Eq. (3.3.13)).

The core lemma is precise enough for our purposes, but the true bounds and premises may be much better. On the one hand, the necessity of the size restrictions on D, K, M is uncertain, as is the role of β. On the other hand, a factor of 4 in the inequalities in Lemma 3.3.12 is a consequence of switching $\vec{\gamma}$ to $\vec{\gamma}'$ (and relying on Lemma A.4.2 to "shrink" the randomness space) instead of analyzing the distribution of the sum S more directly.

3.4. Sharp$_{\mathsf{GS}}$: Batching and Group Switching

In this section, we present the optimized Σ-protocol for showing the decomposition in the DLOG setting, introduce group switching, and show how to perform efficient proofs for batches of integers.

3.4.1. Parameters

Here, we give an overview of all the used parameters in Sharp$_{\mathsf{GS}}$. Let $N \in \mathbb{N}$ be the number of integers $x_1, \ldots, x_N$ in the ranges $[0, B_i]$. In the following, we fix $B = B_i$ for simplicity. Let R be the number of repetitions of the proof and $[0, \Gamma]$ be the challenge set. Generally, we have $R = \lceil \lambda / \log(\Gamma + 1) \rceil$ unless lower soundness than λ bits is satisfactory. We will need to mask values $x \in [0, B\Gamma]$ and values $r \in [0, S\Gamma]$ (where S is defined below) with masking algorithm $\mathsf{mask}_x, \mathsf{mask}_r$, masking randomness distribution R_x, R_r, masking overhead L_x, L_r and masking abort probability $\mathsf{p_x}, \mathsf{p_r}$ respectively. Let $p \geq 2(B\Gamma^2 + 1)L_x$ and $q \geq 18((B\Gamma + 1)L_x)^2$. We use MPed commitments with hiding parameter S in groups $\mathbb{G}_{\mathsf{com}}$ and $\mathbb{G}_{\mathsf{3sq}}$, with prime order p and q respectively. We fix generators $G_0, G_i, G_{i,j} \xleftarrow{\$} \mathbb{G}_{\mathsf{com}}$ for the commitment key $ck_{\mathbb{G}_{\mathsf{com}}}$ and $H_0, H_i \xleftarrow{\$} \mathbb{G}_{\mathsf{3sq}}$ for $ck_{\mathbb{G}_{\mathsf{3sq}}}$, where $i \in [1, N]$ and $j \in [1, 3]$. Let Hash be a collision resistant hash function with output size 2λ bits. The CRS is $crs = (ck_{\mathbb{G}_{\mathsf{com}}}, ck_{\mathbb{G}_{\mathsf{3sq}}})$.

3.4.2. Scheme Overview

The Σ-protocol Sharp$_{\mathsf{GS}}$ is described in Algorithm 1. The prover receives the witnesses $x_i \in [0, B]$ and $r_x \in [0, S]$, and the statement $C_x = r_x G_0 + \sum_{i=1}^{N} x_i G_i$ and B as input. Prover and verifier proceed as follows: (1) In the first flow, the prover computes and commits to a decomposition of x_i using MPed in $\mathbb{G}_{\mathsf{com}}$ (Lines 1 and 2). Then, for all repetitions $k \in [1, R]$, she commits to random masks of the witnesses and decomposition in MPed over $\mathbb{G}_{\mathsf{com}}$ (Line 4 to 7) and the garbage terms of the decomposition polynomial (lines 8 to 12). Finally, she sends the commitments to the verifier. (2) In the second flow, the verifier draws a random challenge for each repetition (Line 1) and sends it to the prover. (3) In the third flow, the prover masks the witnesses (multiplied with the challenges) for each repetition and sends the result to the verifier (lines 13 to 18). (4) Finally, the verifier checks whether the linear relation between the commitments and the challenge holds, after recomputing the decomposition polynomial (lines 2 to 8).

Optimizations. We use uniform rejection sampling for the masking (instead of Gaussian rejection sampling in CKLR). This reduces the masking overhead in our setting. As in CKLR, the prover can avoid sending the commitments $\mathcal{D} = (D_{k,x}, D_{k,y}, D_{k,*})_{k=1}^{R}$ by replacing the output $\mathcal{D}$ in the first flow with a hash $\Delta \longleftarrow \mathsf{Hash}(\mathcal{D})$. Then, the verifier can recompute $\mathcal{D}$ in the verification and check whether the hash matches. Applying the Fiat-Shamir transformation yields a non-interactive range proof.

3.4.3. Security and Correctness

Non-abort probability. With R repetitions, the probability of the honest prover *not* aborting (due to masking) is lower-bounded by $[(1 - \mathsf{p_r})^3 \cdot (1 - \mathsf{p_x})^{4N}]^R$.

Security. Sharp$_{\mathsf{GS}}$ proofs satisfy correctness, non-abort SHVZK and relaxed soundness. Intuitively, the verifier is convinced that the committed value has a *unique* rational representative in the range $[-\frac{1}{4B}, B + \frac{1}{4B}]_{\mathbb{Q}}$, formalized in Theorem 3.4.1 below. Note that with the *four* square decomposition, we obtain exact range membership in $[0, B]$, in exchange for slightly increasing proof size (see Section 3.6.1.2).

Theorem 3.4.1. *The scheme* Sharp$_{\mathsf{GS}}$ *has correctness error at most* $1-[(1-\mathsf{p_r})^3\cdot(1-\mathsf{p_x})^N]^R$. *It is non-abort SHVZK under the SEI assumption in* $\mathbb{G}_{\mathsf{com}}$ *and* $\mathbb{G}_{\mathsf{3sq}}$. *If* $2(B\Gamma^2+1)L < p$ *and* $18K^2 < q$ *with* $K = (B\Gamma + 1)L$, *then* Sharp$_{\mathsf{GS}}$ *has relaxed soundness under the DLOG and SEI assumptions in* $\mathbb{G}_{\mathsf{com}}$ *and* $\mathbb{G}_{\mathsf{3sq}}$ *with knowledge error* $(\frac{2}{\Gamma+1})^R$ *for the relation* $\mathcal{R}_{\mathsf{Ext}} = \{((x_i)_{i=1}^N, r_x) \colon C_x = r_x G_0 + \sum_{i=1}^N x_i G_i \wedge [x_i]_{\mathbb{Q}} \in [-\frac{1}{4B}, B + \frac{1}{4B}]_{\mathbb{Q}_{K,\Gamma}}\}$. *To be precise, we consider the S-bounded SEI assumption in* $\mathbb{G}_{\mathsf{com}}$ *and* $\mathbb{G}_{\mathsf{3sq}}$. *Moreover, in* $\mathcal{R}_{\mathsf{Ext}}$ *all* $[x_i]_{\mathbb{Q}}$ *have a common denominator* $d \in [1, \Gamma]$.

Security proof, outline. Here, we only sketch the proof of security and the relaxed soundness guarantee. We refer to Appendix A.5.1 for details. (The proof is given for the Sharp$_{\mathsf{GS}}$ with all optimizations.) Informally, the committed x_i are guaranteed to have rational representatives in $[-\frac{1}{4B}, B + \frac{1}{4B}]_{\mathbb{Q}_{K,\Gamma}}$, where the numerator and denominator is bounded by $K = (B\Gamma + 1)L$ and Γ respectively.

Since either mask aborts or the z's lie within a predetermined range, correctness follows easily. Also, we can simulate a valid transcript of the proof for statement (C_x, B) by first sampling the challenge and then computing a transcript starting from the last flow. For this, we replace each witness w in the masking $\mathsf{mask}(\gamma w, \widetilde{w})$ with 0 (where $\widetilde{w}$ is the used mask) which affects the distribution only by $\varepsilon_{\mathsf{mask}} = 0$ (see Section 3.2.4). If any masking aborts, the simulator returns $\perp$. Thus, the scheme is non-abort SHVZK under the SEI assumption (for hiding commitments). For the soundness proof, we show 3-special soundness, i.e. extraction from 3 related transcripts. First, we extract the commitments (with a standard argument). Second, we verify that the three square decomposition holds over $\mathbb{Z}_q$ for the extracted x_is and infer that $[x_i]_{\mathbb{Q}} \in [-\frac{1}{4B}, B + \frac{1}{4B}]_{\mathbb{Q}}$ using Lemma A.2.2. The switch between groups requires special care, as the rings $\mathbb{Z}_p$ and $\mathbb{Z}_q$ are "algebraically incompatible". But the shortness of the extracted values suffices to show that the three square decomposition over $\mathbb{Z}_q$ implies non-negativity for the rational representative committed over $\mathbb{Z}_p$.

Algorithm 1 Sharp$_{\mathrm{GS}}$

Prover$(C_x, B, r_x, \{x_i\}_{i=1}^N)$ Verifier(C_x, B)

```
1:  Compute $y_{i,j}$ s.t. $4x_i(B - x_i) + 1 = \sum_{j=1}^3 y_{i,j}^2$ for $i \in [1, N]$
2:  Set $C_y = r_y G_0 + \sum_{i=1}^N \sum_{j=1}^3 y_{i,j} G_{i,j}$ for $r_y \xleftarrow{\$} [0, S]$
3:  for all $k \in [1, R]$ do
4:      Set $\widetilde{r}_{k,x}, \widetilde{r}_{k,y} \xleftarrow{\$} \mathsf{R}_r$    ▷ Opening
5:      Set $\widetilde{x}_{k,i}, \widetilde{y}_{k,i,j} \xleftarrow{\$} \mathsf{R}_x$ for $i \in [1, N], j \in [1, 3]$
6:      Set $D_{k,x} = \widetilde{r}_{k,x} G_0 + \sum_{i=1}^N \widetilde{x}_{k,i} G_i$
7:      Set $D_{k,y} = \widetilde{r}_{k,y} G_0 + \sum_{i=1}^N \sum_{j=1}^3 \widetilde{y}_{k,i,j} G_{i,j}$
8:      Set $r_k^* \xleftarrow{\$} [0, S]$ and $\widetilde{r}_k^* \xleftarrow{\$} \mathsf{R}_r$    ▷ Decomposition
9:      Set $\alpha_{1,k,i}^* = 4\widetilde{x}_{k,i} B - 8x_i \widetilde{x}_{k,i} - 2\sum_{j \in [1,3]} y_{i,j} \widetilde{y}_{k,i,j}$ for $i \in [1, N]$
10:     Set $\alpha_{0,k,i}^* = -(4\widetilde{x}_{k,i}^2 + \sum_{j \in [1,3]} \widetilde{y}_{k,i,j}^2)$ for $i \in [1, N]$
11:     Set $C_{k,*} = r_k^* H_0 + \sum_{i=1}^N \alpha_{1,k,i}^* H_i$
12:     Set $D_{k,*} = \widetilde{r}_k^* H_0 + \sum_{i=1}^N \alpha_{0,k,i}^* H_i$
```

$$\xrightarrow{C_y,\ \{C_{k,*}, D_{k,x}, D_{k,y}, D_{k,*}\}_{k=1}^R}$$

```
1:  $\gamma_k \xleftarrow{\$} [0, \Gamma]$ for $k \in [1, R]$    ▷ Challenge
```

$$\xleftarrow{\{\gamma_k\}_{k=1}^R}$$

```
13: for all $k \in [1, R], i \in [1, N], j \in [1, 3]$ do
14:     Set $z_{k,i} = \mathsf{mask}_x(\gamma_k \cdot x_i, \widetilde{x}_{k,i}), z_{k,i,j} = \mathsf{mask}_x(\gamma_k \cdot y_{i,j}, \widetilde{y}_{k,i,j})$
15:     Set $t_{k,x} = \mathsf{mask}_r(\gamma_k r_x, \widetilde{r}_{k,x}), t_{k,y} = \mathsf{mask}_r(\gamma_k \cdot r_y, \widetilde{r}_{k,y})$
16:     Set $t_k^* = \mathsf{mask}_r(\gamma_k \cdot r_k^*, \widetilde{r}_k^*)$
17: if any $z_{k,i}$, $t_{k,x}$ or $t_k^*$ is $\perp$ then
18:     abort    ▷ Masking failed
```

$$\xrightarrow{\{z_{k,i,j}, z_{k,i}, t_{k,x}, t_{k,y}, t_k^*\}_{k \in [1,R], i \in [1,N], j \in [1,3]}}$$

```
2:  for all $k \in [1, R]$ do
3:      Check $D_{k,x} + \gamma_k C_x = t_{k,x} G_0 + \sum_{i=1}^N z_{k,i} G_i$
4:      Check $D_{k,y} + \gamma_k C_y = t_{k,y} G_0 + \sum_{i=1}^N \sum_{j=1}^3 z_{k,i,j} G_{i,j}$
5:      Set $f_{k,i}^* = 4z_{k,i}(\gamma_k B - z_{k,i}) + \gamma_k^2 - \sum_{j=1}^3 z_{k,i,j}^2$
6:      Check $D_{k,*} + \gamma_k C_{k,*} = t_k^* H_0 + \sum_{i=1}^N f_{k,i}^* H_i$
7:      Check $z_{k,i}, z_{k,i,j} \in [0, (B\Gamma + 1)L_x]$ for $i \in [1, N], j \in [1, 3]$
8:  return 1 iff all checks succeed
```

3.5. $\mathsf{Sharp}^{\mathsf{Po}}_{\mathsf{SO}}$: Improved Proof of Short Opening

We present $\mathsf{Sharp}^{\mathsf{Po}}_{\mathsf{SO}}$, which is based on $\mathsf{Sharp}_{\mathsf{GS}}$ but uses a (batch) shortness test to separate PoSO and PoDec, and to reduce costs of "internal" repetitions.

3.5.1. Parameters

The groups $\mathbb{G}_{\mathrm{com}}$ and $\mathbb{G}_{3\mathrm{sq}}$, and parameters B, Γ, N, and S, are identical to $\mathsf{Sharp}_{\mathsf{GS}}$ (cf. Section 3.4.1). The commitment key ck_{com} is augmented by additional elements $\widetilde{G}_j \xleftarrow{\$} \mathbb{G}_{\mathrm{com}}$ for $j \in [1, R]$. For simplicity, we define $\widehat{\Gamma} := (\Gamma + 1)^R - 1$ (the size of "large" challenge), and require that $\widehat{\Gamma} \leq p$.[9]

More concretely, we consider a *small group* $\mathbb{G}_{\mathrm{com}}$ of order p and a *large group* $\mathbb{G}_{3\mathrm{sq}}$ (which may be equal) of order q. Let B be a *range bound* and let Γ be a bound for the *challenge sizes*. Let N be the *batch size*, i.e. the number of committed x_i whose range membership is to be proven. Let R be the number of "internal repetitions" of the Batch-PoSO. Let $\widehat{\Gamma} := (\Gamma + 1)^R - 1$ be the size of "large" challenges, and assume that $\widehat{\Gamma} \leq p$.

Commitment key setup. The commitment key is $ck = (ck_{\mathrm{com}}, ck_{3\mathrm{sq}})$, where

- $ck_{\mathrm{com}} = (\{G_i\}_{i \in [0,N]}, \{G_{i,j}\}_{i \in [1,N], j \in [1,3]}, \{\widetilde{G}_j\}_{j \in [1,R]})$, where $G_i, G_{i,j}, \widetilde{G}_j \xleftarrow{\$} \mathbb{G}_{\mathrm{com}}$.
- $ck_{3\mathrm{sq}} = (\{H_i\}_{i \in [0,N]})$, where $H_i \xleftarrow{\$} \mathbb{G}_{3\mathrm{sq}}$.

Analogous to $\mathsf{Sharp}_{\mathsf{GS}}$, the elements G_0 and H_0 are used for random masking terms of the commitment, the elements $G_1, \dots, G_N$ are used to commit to x_i, and $G_{i,1}, \dots, G_{i,3}$ are used for the 3-square decomposition $y_{i,1}, \dots, y_{i,3}$ of $1 + 4x_i(B - x_i)$, and the elements $H_1, \dots, H_N$ are used to commit to the garbage terms for linearization of the square decomposition proof. The new elements $\widetilde{G}_1, \dots, \widetilde{G}_R$ are for Batch-PoSO masks μ_j,

Masking and mask sizes. For simplicity, we fix a single masking overhead L for all masks. Logically, some masks must be short due to shortness checks, while other masks only hide the value and shortness is used to reduce communication. The latter may be drawn uniformly from $\mathbb{Z}_p$ as well. In $\mathsf{Sharp}_{\mathsf{GS}}$, L_x was the former, L_r the latter type. In $\mathsf{Sharp}^{\mathsf{Po}}_{\mathsf{SO}}$, we have following masking behaviour:

- $\mathsf{R}_{poso} = [0, (V_{poso} + 1)L]$, where $V_{poso} = 4NB\Gamma$ must be short.
- For $z \in \{x, \mu, r, r^*\}$, R_z need only hide the value, so $\mathsf{mask}_z(v, m)$ is *computed modulo p* (resp. q). If $\mathsf{R}_z = \mathbb{Z}_p$ (resp. $\mathbb{Z}_q$), mask_z never aborts.
- For $z \in \{x, \mu, r\}$, we set $\mathsf{R}_z = [0, \min(p - 1, (V_z + 1)L)]$, where $V_x = B\widehat{\Gamma}$, $V_r = S$, and $V_\mu = \mathsf{R}_{poso} \cdot \widehat{\Gamma} L$that is, $V_\mu = (V_{poso} + 1)L \cdot \widehat{\Gamma} L$. And we set $\mathsf{R}_{r^*} = [0, \min(q - 1, (V_{r^*} + 1)L)]$ where $V_{r^*} = S$.
- If $\mathbb{G}_{\mathrm{com}} = \mathbb{G}_{3\mathrm{sq}}$, then typically $\mathsf{R}_r = \mathsf{R}_{r^*} = \mathsf{R}_\mu = \mathbb{Z}_p$.

[9] Since the maximal challenge set for a scalar challenge is $[0, p - 1] = \mathbb{Z}_p$, increasing the challenge set would require repetitions in "Phase 2", which is trivially implemented but completely unnecessary for our instantiations.

3.5.2. Scheme Overview

The difference between Sharp_{GS} and $\text{Sharp}^{\text{Po}}_{\text{SO}}$ is the use of the Batch-PoSO. Again, to simplify we only consider one range $[0, B]$ for all x_i. It will be evident how to generalize to independent ranges $x_i \in [0, B_i]$.

The scheme is defined in Algorithms 2 and 3. It is a 5-move protocol which effectively consists of 2 phases: In Phase 1, the prover commits to the 3-square decompositions (and masks μ_k). Then, k parallel random affine shortness tests are run on committed values. In Phase 2, the prover proves that it has correctly answered the shortness test, and that the 3-square decomposition holds modulo q. Thus, Phase 2 is very similar to Sharp_{GS}, except, it uses a large challenge space $[0, \widehat{\Gamma}]$, so no repetitions are required.

Algorithm 2 $\text{Sharp}^{\text{Po}}_{\text{SO}}$– Phase 1

```
Prover($C_x, B, r_x, \{x_i\}_{i=1}^N$)                                Verifier($C_x, B$)

1: Compute $4x_i(B - x_i) + 1 = \sum_{j=1}^3 y_{i,j}^2$ for $i \in [1, N]$
2: Set $r_y \xleftarrow{\$} [0, S]$ and $\mu_1, \ldots, \mu_R \xleftarrow{\$} \mathsf{R}_{poso}$
3: Set $C_y = r_y G_0 + \sum_{i=1}^N \sum_{j=1}^3 y_{i,j} G_{i,j} + \sum_{k=1}^R \mu_k \widetilde{G}_k$

                         $\xrightarrow{C_y}$

                         1: Sample $\gamma_{i,j}^{(k)} \xleftarrow{\$} [0, \Gamma]$ for $i \in [1, N]$, $j \in [0, 3]$, $k \in [1, R]$

                         $\xleftarrow{\{\gamma_{i,j}^{(k)}\}_{i,j,k}}$

4: Let $y_{i,0} := x_i$
5: Set $\zeta_k := \mathsf{mask}_{poso}(\sum_{i=1}^N \sum_{j=0}^3 \gamma_{i,j}^{(k)} y_{i,j}, \mu_k)$ for $k \in [1, R]$
6: if any $\zeta_k$ is $\perp$ then
7:     abort                                         ▷ Masking Failed

                         $\xrightarrow{\{\zeta_k\}_{k \in [1,R]}}$

                         2: if any $\zeta_k \notin [0, (4NB\Gamma + 1)L]$ then
                         3:     return 0                              ▷ PoSO rejected

     Run Phase 2: Proof of consistency of $\zeta_k$ and 3-square decomposition (see Algorithm 3)
```

3.5.3. Security and Correctness

Non-abort probability. With R "internal" repetitions, the number of masking operations are R in Phase 1. In Phase 2, we have $4N$ for x_i and $y_{i,j}$, again R for μ_k, and 3 masks for r_x, r_y, r^*. Thus, the probability of the honest prover *not* aborting (due to masking) is lower-bounded by $(1 - \frac{1}{L})^{2R+4N+3}$.

Algorithm 3 $\text{Sharp}_{\text{SO}}^{\text{Po}}$– Phase 2

```
                    After Phase 1 (shortness proof, see Algorithm 2)
 8: Set $\widetilde{r}_x, \widetilde{r}_y \xleftarrow{\$} \mathsf{R}_r$
 9: Set $\widetilde{x}_i, \widetilde{y}_{i,j} \xleftarrow{\$} \mathsf{R}_x$ for $i \in [1,N], j \in [1,3]$
10: Set $\widetilde{\mu}_k \xleftarrow{\$} \mathsf{R}_\mu$ for $k \in [1,R]$                                  ▷ PoSO
11: Set $d_k = \sum_{i=1}^N \sum_{j=0}^3 \widetilde{y}_{i,j}\gamma_{i,j}^{(k)} + \widetilde{\mu}_k$ for $k = 1,\ldots,R$          ▷ PoSO
12: Set $D_x = \widetilde{r}_x G_0 + \sum_{i=1}^N \widetilde{x}_i G_i$
13: Set $D_y = \widetilde{r}_y G_0 + \sum_{i=1}^N \sum_{j=1}^3 \widetilde{y}_{i,j} G_{i,j} + \sum_{k=1}^R \widetilde{\mu}_k \widetilde{G}_k$
14: Set $r^* \xleftarrow{\$} [0,S]$ and $\widetilde{r}^* \xleftarrow{\$} \mathsf{R}_{r^*}$
15: Set $\alpha_{1,i}^* = 4\widetilde{x}_i B - 8x_i\widetilde{x}_i - 2\sum_{j\in[1,3]} y_{i,j}\widetilde{y}_{i,j}$ for $i \in [1,N]$
16: Set $\alpha_{0,i}^* = -(4\widetilde{x}_i^2 + \sum_{j\in[1,3]} \widetilde{y}_{i,j}^2)$ for $i \in [1,N]$
17: Set $C_* = r^* H_0 + \sum_{i=1}^N \alpha_{1,i}^* H_i$
18: Set $D_* = \widetilde{r}^* H_0 + \sum_{i=1}^N \alpha_{0,i}^* H_i$

                    $\xrightarrow{C_*, D_x, D_y, D_*, \{d_k\}_{k=1}^R}$

                     4: $\gamma \xleftarrow{\$} [0, (\Gamma+1)^R - 1) \subseteq \mathbb{Z}_p$                  ▷ Large challenge

                    $\xleftarrow{\gamma}$

19: for all $i \in [1,N], j \in [1,3], k \in [1,R]$ do
20:     Set $z_i = \mathsf{mask}_x(\gamma \cdot x_i, \widetilde{x}_i)$ and $z_{i,j} = \mathsf{mask}_x(\gamma \cdot y_{i,j}, \widetilde{y}_{i,j})$
21:     Set $t_x = \mathsf{mask}_r(\gamma \cdot r_x, \widetilde{r}_x)$ and $t_y = \mathsf{mask}_r(\gamma \cdot r_y, \widetilde{r}_y)$
22:     Set $t^* = \mathsf{mask}_r(\gamma \cdot r^*, \widetilde{r}^*)$
23:     Set $\tau_k = \mathsf{mask}_\mu(\gamma \cdot \mu_k, \widetilde{\mu}_k)$                        ▷ PoSO
24: if any $z_i$, $z_{i,j}$, $t_x$, $t_y$ $t^*$, $\tau_k$ is $\perp$ then
25:     abort                                                   ▷ Masking failed

                    $\xrightarrow{\{z_i\}_{i\in[1,N]}, \{z_{i,j}\}_{i\in[1,N], j\in[1,3]}, t_x, t_y, t^*, \{\tau_k\}_{k\in[1,R]}}$

                     5: Compute $F_x = -\gamma C_x + t_x G_0 + \sum_{i=1}^N z_i G_i$
                     6: Compute $F_y = -\gamma C_y + t_y G_0 + \sum_{i=1}^N \sum_{j=1}^3 z_{i,j} G_{i,j} + \sum_{k=1}^R \tau_k \widetilde{G}_k$
                     7: Let $z_{i,0} := z_i$
                     8: Set $f_k = -\gamma\zeta_k + \sum_{i=1}^N \sum_{j=0}^3 z_{i,j}\gamma_{i,j}^{(k)} + \tau_k$ for $k \in [1,R]$          ▷ PoSO
                     9: Compute $f_i^* = 4z_i(\gamma B - z_i) + \gamma^2 - \sum_{j=1}^3 z_{i,j}^2$ for $i \in [1,N]$
                    10: Recompute $F_* = -\gamma C_* + t^* H_0 + \sum_{i=1}^N f_i^* H_i$
                    11: if $F_x = D_x$, $F_y = D_y$, $F_* = D_*$, and $f_k = d_k$ for $k \in [1,R]$ then
                    12:     return 1
                    13: else return 0
```

Security. The security guarantee of $\mathsf{Sharp}^{\mathsf{Po}}_{\mathsf{SO}}$ is almost the same as that of $\mathsf{Sharp}_{\mathsf{GS}}$, except for a small tightness loss due to the weaker (provable) guarantees of the shortness test (Theorem 3.3.3).

Theorem 3.5.1. *The scheme* $\mathsf{Sharp}^{\mathsf{Po}}_{\mathsf{SO}}$ *has correctness error at most* $1-(1-\frac{1}{L})^{2R+4N+3}$. *It is non-abort SHVZK under the SEI assumption in* $\mathbb{G}_{\mathsf{com}}$ *and* $\mathbb{G}_{3\mathsf{sq}}$. *Let* $K' = (1+2\beta)K$ *where* $K = (B\Gamma+1)L$ *and* $\beta = \min(4N, \mathsf{primlmin}(\Gamma+1))$. *If* $18(K')^2 < q$ *and* $2(\Gamma+1)^2K' < p$ *and* $(\Gamma+1)^R - 1 < p$, *then* $\mathsf{Sharp}^{\mathsf{Po}}_{\mathsf{SO}}$ *has relaxed soundness under the DLOG and SEI assumptions in* $\mathbb{G}_{\mathsf{com}}$ *and* $\mathbb{G}_{3\mathsf{sq}}$ *with knowledge error* $\frac{2+8^R}{(\Gamma+1)^R}$ *for the relation* $\mathcal{R}_{\mathsf{Ext}} = \{((x_i)_{i=1}^N, r_x)\colon C_x = r_xG_0 + \sum_{i=1}^N x_iG_i \wedge [x_i]_{\mathbb{Q}} \in [-\frac{1}{4B}, B+\frac{1}{4B}]_{\mathbb{Q}_{K',\Gamma}}\}$. *To be precise, we consider the* S*-bounded SEI assumption in* $\mathbb{G}_{\mathsf{com}}$ *and* $\mathbb{G}_{3\mathsf{sq}}$. *Moreover, in* $\mathcal{R}_{\mathsf{Ext}}$ *all* $[x_i]_{\mathbb{Q}}$ *have a common denominator* $d \in [1,\Gamma]$.

Security proof, outline. The proof of correctness and non-abort SHVZK for $\mathsf{Sharp}^{\mathsf{Po}}_{\mathsf{SO}}$ are completely analogous to the respective proofs for $\mathsf{Sharp}_{\mathsf{GS}}$.

The ideas behind the soundness proof of Theorem 3.5.1 are quite straightforward. It proceeds by dealing with the two phases separately. First, observe that Phase 2 is effectively a Σ-protocol for the statement which was completed in Phase 1, i.e. that C_x resp. C_y are commitments to the x_i's resp. auxiliary values $y_{i,j}$ and μ_k, the answers ζ_k of a random affine shortness test are correct, and the 3-square decomposition holds. Indeed, Phase 2 is 3-special sound, i.e. given 3 accepting transcripts identical up until the challenge message γ for 3 distinct challenges, one can extract openings to the commitments which satisfy the relation (or the binding property is broken). Thus, as a first step, one can replace Phase 2 with an extractor with knowledge error $2/(\Gamma+1)^R$.

Next, one needs to argue that the x_i and $y_{i,j}$ are short (from above, we know that they satisfy the 3-square decomposition). This does not follow from (3 transcripts for) Phase 2 alone. Intuitively, if the "shortness test" used has soundness error δ_{snd}, then if any x_i, $y_{i,j}$ is not short, the probability that the verifier accepts is at most δ_{snd}^R. More precisely, if there is no $d \in [1,\Gamma]$ such that $dx_i, dy_{i,j} \in [-K', K']_{\mathbb{Z}_p}$ for all i, j, then the shortness test accepts with probability at most δ_{snd}. However, there is a gap: Our commitment is only computationally binding, so, by breaking the commitment, the adversary might win with probability ε (much) higher than δ_{snd}^R. Fortunately, to win with probability $\varepsilon > \delta_{\mathsf{snd}}^R$, the adversary *must* break the binding property. Thus, except with probability δ_{snd}^R, one obtains a binding break from such an adversary in expected time (by rewinding until $\mathcal{A}$ succeeds again). Overall, this proves the soundness claim of Theorem 3.5.1.

3.5.4. Trade-offs and Optimizations

Reducing communication. As with $\mathsf{Sharp}_{\mathsf{GS}}$, hashing can reduce the communication in Phase 2 of the protocol. Namely, the final verification step in Phase 2 computes F_x, F_*, $\{f_k\}_{k\in[1,R]}$, and checks if they are equal to D_x, D_*, $\{d_k\}_{k\in[1,R]}$. This check can be compressed by using a collision resistant hash function H, and having the prover send $D_H = H(D_x, D_y, D_*, \{d_k\}_{k\in[1,R]})$ instead. The verification now checks $D_H = F_H$, where $F_H \coloneqq H(F_x, F_*, \{f_k\}_{k\in[1,R]})$. It is easy to see that the protocol remains secure if H is collision resistant. Also, since Phase 2 is effectively independent of Phase 1, it may be exchanged with other suitable (succinct) argument systems. This is discussed in a later paragraph.

Fiat–Shamir transformation. $\mathsf{Sharp}_{\mathsf{SO}}^{\mathsf{Po}}$ is public-coin and the Fiat–Shamir transformation is applicable. This yields a *non-interactive* zero-knowledge argument. If the (hash) function is modelled as a random oracle, the resulting scheme is provably secure in the ROM, although there is a security loss (in the number of random oracle queries).

As $\mathsf{Sharp}_{\mathsf{SO}}^{\mathsf{Po}}$ is not a usual Σ-protocol, nor special sound (with sensible parameters), well-known extraction techniques are not directly applicable. However, the reasoning for the security of the Fiat–Shamir transformation of multi-round special sound protocols in recent works [AFK21; Wik21] should be applicable to our setting. After all, the step in Phase 1 is not particularly involved, and we have a property akin to special soundness there: If a second transcript is necessary (due to inconsistent witness extracted in Phase 2), then a uniformly *random* accepting transcript (with same message) will, with high probability, lead to a non-trivial DLOG relation.

Proving non-negativity. As with CKLR proofs [CKLR21b], it is possible to only prove $x \geq 0$ instead of $x \in [0, B]$. Namely, using $1 + 4x = \sum_{i=1}^{3} y_i^2$ shows $x \geq -1/4$, and using the four square decomposition shows $x \geq 0$. This is of interest if the upper bound B is "unreachable" or otherwise not of interest. However, an upper bound B for x is still required (and must not be too large), as it determines the size of the masks and the verifier's size checks as before (since wrap-around must still be prevented). Moreover, B is the maximal value for which zero-knowledge guarantees hold; the larger $x > B$ becomes, the more zero-knowledge degrades. This optimization applies to $\mathsf{Sharp}_{\mathsf{GS}}$ and $\mathsf{Sharp}_{\mathsf{SO}}^{\mathsf{Po}}$.

Standard Soundness and higher knowledge error. It is easy to see that RAST with uniform distribution over $\{0, 1\}^N$ is fractionally $(NBL, 1)$-sound with error $1/2$. In this case, $\mathsf{Sharp}_{\mathsf{SO}}^{\mathsf{Po}}$ has standard soundness with knowledge error $\kappa = 2^{-R}$, and R repetitions require approximately $2R \cdot \log(NBL)$ bits communication overhead. This trade-off is especially interesting if high knowledge error is acceptable. for example, a statistical knowledge error $\kappa = 2^{-40} + \mathsf{negl}$ in interactive settings[10] is a common choice, and in application to anonymous credentials may be considered acceptable.

By using the Fiat–Shamir transformation on Phase 2 (with $\widehat{\Gamma} = 2^{\lambda} - 1$), an interactive 3-move protocol can be obtained.[11] The trade-off is also useful if batch size N is huge (hence amortized cost to achieve standard soundness is small). In that case, exchanging Phase 2 is also of interest.

Exchanging phase 2. Since Phase 2 is effectively independent of Phase 1, i.e. the shortness test, it may be exchanged with other suitable (succinct) argument systems. This is especially interesting to reduce overall communication. As only knowledge of openings and simple quadratic equations are proven, generic proof systems which target R1CS over $\mathbb{Z}_p$ (e.g. Bulletproofs [BCC+16; BBB+18]) or general quadratic or polynomial equations over $\mathbb{Z}_p$ (e.g. [HKR19a; BG18]) can be used as drop-ins.

In fact, the ζ_k could also be committed to and proven to be computed correctly and that they lie within $[0, K]$; if done in zero-knowledge, this makes the masking terms μ_k superfluous, improving the soundness error of the shortness test. However, a (standard sound) range proof is needed to check $\zeta_k \in [0, 4NB\Gamma]$, and the proof system must now include adaptively chosen commitments and statements, which typically is not a problem for commit-and-prove-based proof systems, but it does

[10] In this case, the communication overhead is reasonable and computational efficiency remains excellent. For 128 repetitions, the communication overhead becomes noticeable. See Table A.2 for concrete size estimates.

[11] We stress that high knowledge error, e.g. 2^{-40}, only makes sense in interactive settings. Fiat–Shamir transformations are trivial (and cheap) to break in this regime.

slightly increase proof size and round complexity. Considering the number of variables for adding the necessary constraints for the (bit-decomposition) range proof for ζ_k, and using binary challenges in the Batch-PoSO (to obtain standard soundness) with $R = 128$ repetitions for security, we obtain a break-even in the number of (auxiliary) variables at about $N = 170$ with naive R1CS-type constraints (and $N = 160$ for quadratic constraints). Overall, for large enough batches sizes, this approach may be of interest. See Example 3.5.2 for more details.

Example 3.5.2 (Using a succinct Phase 2). As discussed in Section 3.5.4, it is possible to adapt suitable succinct arguments which follow a commit-then-prove strategy which allow multiple commitment steps and an adaptive choice of the final statement. The upside is, that the 3-square decomposition requires fewer auxiliary variables (compared to bit-decomposition). The downside is, that an overhead which is almost independent of the batch size must be paid (namely, the R repetitions). We discuss and roughly quantify this trade-off, where we use the PoSO with binary challenges to achieve standard soundness. For concreteness, consider the Bulletproofs variant [HKR19a], which allows proving quadratic equations over committed variables (instead of the weaker R1CS-type equations). We set $B = 2^{64} - 1$ and $R = 128$ (and $\Gamma = 1$).

The protocol with exchanged Phase 2 works as follows:

1. (Phase 1) Commit not only to x_i, but also to the square decomposition $y_{i,j}$ and masks μ_k for $k = 1, \dots, R$.
2. Receive the PoSO challenges and responds with ζ_k.
3. (Phase 2) Both prover and verifier adapt the statement by including the linear check constraints (for $k = 1, \dots, R$) for the PoSO, similar to Phase 2 of $\mathsf{Sharp}^{\mathsf{Po}}_{\mathsf{SO}}$.

With this approach, sending $\{\zeta_k\}_{k=1}^{R}$ significantly increases the proof size (namely, by R elements of $\log((4N\Gamma B + 1)L)$ bits each). But only 4 variables per range are used (x_i and 3 auxiliary variables $\{y_{i,j}\}_j$), whereas 64 variables are required per bit-decomposition.

A more complex approach enables smaller proofs, but requires an adaptive commitment and statement:

1. Commit not only to x_i, but also to the square decomposition $y_{i,j}$.
2. Receive the PoSO challenges and *commit* to ζ_k for $k \in [1, R]$.
3. Both prover and verifier adapt the statement by including the linear check constraint (for $k = 1, \dots, R$) for the PoSO, and a (bit-decompostion) range proof for $\zeta_k \in [0, 4N\Gamma B]$.

An advantage is, that no masks are necessary, but we now require an adaptive commitment to ζ_k which still slightly increases proof size. For the normal bit-decomposition of all x_i, one needs $N \log(B + 1)$ variables and quadratic constraints. With this approach, one needs $4N + R \cdot \log(4N\Gamma B + 1)$ variables and quadratic constraint (plus a suitable commit-and-prove system). For $R = \lambda = 128$ and $B = 2^{64} - 1$ the break-even point in terms of variables and constraints is at about $N = 160$, and at $N = 2048$ we observe an over 7-fold reduction in terms of variables and constraints, which tends to 16 as N grows. For R1CS-type constraints (now using 8 instead of 4 variables), we observe break-even at about $N = 170$ and an almost 5-fold reduction at $N = 2048$ which tends to 8 as N grows.

3.6. Soundness Guarantees and Hidden Order Augmentation

We provide some insights into the consequences of relaxed soundness and the use of hidden order groups in that context. Further discussions can be found in Appendix A.2 and Appendix A.3 respectively.

3.6.1. Remarks on Relaxed Soundness

The relaxed soundness of CKLR-type proofs only ensures that a committed value x is a fraction $x \equiv_p m/d$ with short numerator and denominator, say $x \in \mathbb{Q}_{M,D}$. As we will see, this can be sufficient in important applications, such as anonymous credentials. However, this guarantee is, in general, too weak to allow unchecked homomorphic operations on commitments, e.g. the sum $\sum_{i=1}^{N} \frac{m_i}{d_i}$ of short fractions m_i/d_i need not be short. The main problem is the growth of the common denominator as $d = \mathrm{lcm}(d_1, \ldots, d_N)$, and the numerator grows similarly. Thus, after a few operations, all guarantees on shortness are lost.

3.6.1.1. Cheating with Small Denominators

The use of relaxed soundness is *not* a proof artefact: For small d and m, find $\sum_{j=1}^{3} a_j^2 = d^2 + 4(m-d)m$ and let $x \equiv_p m/d$ and $y_j \equiv_p a_j/d$. This decomposition has a chance of $1/d$ (per repetition, and $1/d^R$ overall) to fool the verifier. In particular, after the Fiat–Shamir transformation, generating proofs for x is efficiently possible if d is not too large.

3.6.1.2. Three Square Decomposition

Our range proofs use the 3-square decomposition and prove membership in $[-\frac{1}{4B}, B + \frac{1}{4B}]_{\mathbb{Q}_{K',\Gamma}}$. To obtain membership in $[0, B]_{\mathbb{Q}_{K',\Gamma}}$ one can either use the 4-square decomposition, or use $\Gamma < 4B$ (perhaps, increasing repetitions), as this ensures that denominators $d \geq 4B$ violate soundness, hence $[0, B]_{\mathbb{Q}_{K',\Gamma}} = [-\frac{1}{4B}, B + \frac{1}{4B}]_{\mathbb{Q}} \cap \mathbb{Q}_{K',\Gamma} = [-\frac{1}{4B}, B + \frac{1}{4B}]_{\mathbb{Q}_{K',\Gamma}}$.

3.6.2. Using Groups of Hidden Order

The problem of denominator growth can be mitigated by resorting to a group $\mathbb{H}$ of hidden order. For $\mathsf{Sharp}_{\mathsf{GS}}$ and $\mathsf{Sharp}^{\mathsf{Po}}_{\mathsf{SO}}$, the approach works as follows: Add a single additional commitment C'_x to all values x_i in $\mathbb{H}$ (using a MPed commitment). Moreover, include a proof of knowledge of opening of C'_x (to the same value as in C_x). This small change allows us to reduce to properties of $\mathbb{H}$ to control the denominator. Using reasonable assumptions, it can be shown that the denominators d_i are of the form $d_i = e^{k_i}$ for $k_i \in \mathbb{N}_0$.

3.6.2.1. Instantiating the Hidden Order Group.

When instantiating $\mathbb{H}$ with suitable class groups of hidden order for which a plausible strengthened 2-fROOT assumption holds, the prover will be bound to dyadic rationals, i.e. x_i of the form $x_i = m_i/2^{k_i}$. This improves the applicability of the range proof significantly, since, even in homomorphic computations, the common denominator d is of the form 2^k with $k \leq \log(\Gamma)$. This restriction already enables the use of homomorphic computations.

When using RSA groups (with trusted setup), the proof provides standard soundness, since the prover is bound to an integer under the 1-fROOT assumption (a.k.a. strong RSA assumption). Interestingly, even without trusted setup, e.g. in cases with a "designated verifier", we sketch how RSA groups enable the use of Sharp proofs (cf. Section 3.7.3).

We refer to Appendix A.3 for a more detailed overview of these "augmented" schemes with an efficiency and security analysis.

3.6.3. Non-Relaxed Soundness from Prior Knowledge

Prior knowledge on the shortness of committed values can "upgrade" the soundness from relaxed to non-relaxed. Namely, suppose for some reason, that you have prior knowledge or the guarantee that the committed value $x \in \mathbb{Z}_p$ is short, i.e. $x \in [-M, M]$. Then its representative in $\mathbb{Q}_{M,D}$ is an integer (namely, $\frac{x}{1}$). Thus, the range proof then directly implies that $x = [x]_{\mathbb{Q}} \in \mathbb{Z}$ is in the desired range $[0, B]_{\mathbb{Q}}$. More formally, we use that $[-\frac{1}{4B}, B + \frac{1}{4B}]_{\mathbb{Q}} \cap \mathbb{Q}_{M,D} \cap \mathbb{Z} = [0, B]_{\mathbb{Q}} \cap \mathbb{Z} = [0, B]_{\mathbb{Z}}$. Note that this reasoning also works for the range proofs from CKLR [CKLR21b].

3.7. Applications

In this section, we show how range proofs with relaxed soundness, such as Sharp (or CKLR), can be used in certain applications, namely as anonymous credentials and anonymous transactions.

3.7.1. Anonymous Credentials

Anonymous credential schemes [Cha90; CL01; Bra00] allow users to obtain credentials from issuing authorities. Later, the user can present this credential to a verifier, without revealing his identity, which is fixed (but hidden via a commitment) in the credential. These credentials can also have attributes, for example a birthdate or a validity date. When showing the credential, the user might need to show that he is older than 18 or that the credential is still valid in a privacy-preserving manner.

Constructions of anonymous credentials typically rely on very efficient special-purpose zero-knowledge proofs. Concretely, most rely on so-called "CL-type" (algebraic) signature schemes, which come with very efficient proofs of knowledge of a signature on committed messages [CL03]. These are used to sign the identity and attributes of a user. To prove that attributes lie in some range, e.g. for age restrictions or a validity date of the credential, range proofs are employed. Thus, range proofs often constitute a significant, if not dominant, part in computation (and communication) in these settings.

Sharp proofs can often be used as an almost drop-in replacement in such settings. Consider the DLOG setting in a group of prime order p.

- When *issuing* the credential, all attribute values are known to the issuer. Assuming suitably small ranges $[0, B] \subseteq [-K, K]$ for valid attributes, the verifier's validity check of attribute values ensures shortness. If $K < p/(4\Gamma)$, then a rational representative m/d of an attribute x must be of the form $m/1$, i.e. x is a short integer. Thus, our range proof will be standard sound for x (see Section 3.6.3).
- In case of *blind issuance* (where identity and attributes remain (partially) hidden), the relaxed soundness of DLOG-based Sharp *may not* suffice (see Section 3.6.1.1). Here, we can use $\text{Sharp}_{\text{RSA}}$ which provides standard soundness, using a trusted public RSA-based setup of the issuer.
- For *showing* the credential, our range proofs can be used if the (blind) issuing phase ensured that the attributes lie within valid ranges, as in that case, our range proof is standard sound (see Section 3.6.3).

The same reasoning applies to so-called *keyed-verification* anonymous credentials [CMZ14], where the issuer and verifiers have a shared secret key, which allows for more efficient protocols (but restricts the use-cases).

Anonymous credentials and their constructions come in many flavours [RVH17; CR19; BL13], and not all rely on prime order groups alone. Some use pairing groups and some use hidden order groups. Nevertheless, it is very likely that in all these settings, our range proofs offer favourable trade-offs when compared to those in use. For example, while hidden order groups allow for three-square decomposition based range proofs, working in prime order groups is typically more efficient in terms of computation and communication. In the pairing-based settings, the approach of [CCs08] allows quite efficient digit-based decompositions. However, operations in pairing-groups are slower, elements are bigger, and for efficiency, [CCs08] needs relatively large (non-transparent) public parameters.

3.7.2. Updatable Anonymous Credentials and BBAs

A line of works [JR16; HHNR17; BBDE19; HKRR20; BEK+20] uses techniques from anonymous credentials in a "non-static" manner to construct *updateable anoymous credentials* or *black-box accumulation (BBA) schemes*, which can be used for electronic payments, ticket systems, incentive systems and more. Most of the schemes feature range proofs as a core component, as these are required to prevent users from spending more than they have. The *(blind) issuing* process is mostly unchanged in comparison to anonymous credentials. The *show* protocol is replaced by (one or more) *update* protocol(s), which modify the user's attributes (e.g. the user's current balance).

Most applications work in the "public balance update" setting, where the user interacts with an operator, and the operator knows the amount Δ by which a user's (hidden) balance v is changed. That is, after the transaction, the balance should be $v + \Delta$, and for security, $v + \Delta \geq 0$ must be ensured. In this "public balance update" setting, our range proofs are again almost drop-in replacements. Namely, if the security proof ensures that the balance v is "small" (i.e. has rational representative $v/1$), then our proof has standard soundness for $v + \Delta \in [0, B]$. Since the security proofs typically prove inductively that, after each operation, the (new) balance v has certain properties (e.g. lies in the range $[0, B]$), the requirement for our proof to be standard sound is easily seen to be satisfied.

Range proofs are so expensive that early works [JR16; HHNR17] consider weakened (security) requirements to achieve practical efficiency. Even in later works [BBDE19; HKRR20; BEK+20], they amount to a large part of (or even dominate) the runtime. Our optimized range proofs greatly improve efficiency.

3.7.3. Anonymous Transactions

Range proofs are often used in privacy-preserving blockchain-based smart contract platforms in order to ensure that the fixed (but hidden) balance of users is non-negative after performing a transfer [Zca; Mon; BAZB20]. This ensures that no user can spend more coins than he owns while preserving privacy. Thus, this is a "secret balance update" setting. Here, we give an overview on the applicability of Sharp in this context and refer to the full version [CGKR22b] for more details.

When a sender with a balance of b coins performs a transfer of a coins to a receiver, she has to guarantee the following: (1) $b - a \geq 0$, i.e. the sender's balance remains non-negative after the transaction and (2) $a \geq 0$, i.e. the sender transfers a non-negative number of coins to the receiver. Often, the values a and b are committed (or fixed via an encryption), and the sender performs two range proofs to show equations (1) and (2). Unfortunately, even an initial shortness guarantee on the committed balances b is not sufficient for relaxed soundness to provide standard guarantees, as the shortness of a cannot be guaranteed this way. Thus, we cannot replace *all* range proofs with Sharp proofs naively (and doing so would lead to concrete attacks). Nevertheless, *some* range proofs can be replaced with Sharp proofs for efficiency improvements.

Furthermore, in the full version [CGKR22b] we sketch how the use of augmented Sharp proofs, with both an additional RSA and class group element, is sufficient to avoid these attacks *without* trusted setup of the RSA modulus. Perhaps surprisingly, we can still leverage the properties of RSA groups in this case.

4. Efficient Zero-Knowledge Arguments in the DLOG setting, Revisited

This chapter is based on [HKR19a] and [HKR19b] (revision 2019-11-21) and large parts of the chapter are rephrased or taken verbatim from these works My contribution in [HKR19a; HKR19b] are the theoretical aspects. Implementations and practical evaluations in Section 4.5 and Appendix B.6 are by Max Hoffmann.

Compared to [HKR19b] (revision 2019-11-21), the work was significantly revised and extended. In particular, (1) we show how a modified choice of challenge distributions enables extraction from a linear number of transcripts (in the witness dimension), see for Section 4.3.3.2 and Corollary 4.4.12; (2) we provide a corresponding lower bound for black-box extraction in Appendix B.5.1; (3) we discuss two candidate short-circuit extraction strategies in Appendix B.9; (4) we simplified and restructured the discussion in Section 4.3; (5) we provide an extended comparison of R1CS and QE in Appendix B.8.

4.1. Introduction

Zero-knowledge arguments (of knowledge) (ZKAoK) allow a party P, the prover, to convince another party V, the verifier, of the truth of a statement (and knowledge of a witness) without revealing any other information. For example, one may prove knowledge of a valid signature on some message, without revealing the signature. The ability to ensure *correctness* without compromising *privacy* makes zero-knowledge arguments a powerful tool, which is ubiquitous in theory and application of cryptography. Since the first *practical* construction of succinct non-interactive arguments of knowledge (SNARK) [GGPR13], and their application to Blockchain and related areas, research in theory and applications of efficient ZKAoKs has progressed significantly, see the works [GGPR13; DFGK14; GMO16; BCC+16; CDG+17; AHIV17; GMNO18; WTs+18; BSCR+18] to name only a few of the earlier works, with many more constructions since then.

In this chapter, we revisit a line of works [Gro09; BCC+16; BBB+18] in the setting of groups of prime order. From an abstract point of view, in terms of [ZkpComref], one part of our work is in the world of ideal linear commitments (ILC). That is, our verifier can do “matrix-vector queries” on a committed value $\boldsymbol{w}$, e.g. request an opening for a matrix-vector product $\boldsymbol{\Gamma w}$. A priori, this is more powerful than other settings like PCP or IOP, where the verifier’s queries are restricted to point or inner-product queries[ZkpComref]. Nonetheless, the ILC-arguments in [Gro09; BCC+16; BBB+18] only work for the language R1CS “natively”, which is also covered by more restricted verifiers. We show that with ILC, one can directly handle *systems of quadratic equations*, of which R1CS is a special case.

Another part of this chapter treats proofs of knowledge of preimages of group homomorphisms. For example, one can prove knowledge of the decryption of an ElGamal ciphertext like this. This does not fit into the ILC setting, hence we do not use the ILC abstractions.

4.1.1. Basic Techniques

We identify and present basic design principles which underly most existing works on efficient zero-knowledge arguments in the group setting.

In the following, we use implicit representation notation for group elements, see Section 4.2. Let us recall (a slight variant of) the standard Σ-Protocol (Σ_{std}) for proving knowledge of a preimage $\boldsymbol{w}$ for $[\boldsymbol{A}]\boldsymbol{w} = [\boldsymbol{t}]$ for $[\boldsymbol{A}] \in \mathbb{G}^{m \times n}$. This proof covers a large class of statements, including dlog relations, knowing the opening of a commitment, and so on. The protocol works as follows:

- Prover: Pick $\boldsymbol{r} \xleftarrow{\$} \mathbb{F}_p^n$, let $[\boldsymbol{a}] := [\boldsymbol{A}]\boldsymbol{r}$, send $[\boldsymbol{a}]$.
- Verifier: Pick and send $\boldsymbol{x} = (x_1, x_2) \xleftarrow{\$} \mathbb{F}_p^2$ (with $x_2 \neq 0$).
- Prover: Send $\boldsymbol{z} := x_1\boldsymbol{w} + x_2\boldsymbol{r}$.
- Verifier: Accept iff $[\boldsymbol{A}]\boldsymbol{z} = x_1[\boldsymbol{t}] + x_2[\boldsymbol{a}]$.

Intuitively, this is zero-knowledge since $\boldsymbol{r}$ completely masks $\boldsymbol{w}$ in $\boldsymbol{z} = x_1\boldsymbol{w}+x_2\boldsymbol{r}$ (since $x_2 \neq 0$), and finding $\boldsymbol{r}$ from $[\boldsymbol{a}]$ is hard. It is extractable, since two linearly independent challenges $\boldsymbol{x}_1, \boldsymbol{x}_2$ with answers $\boldsymbol{z}_1, \boldsymbol{z}_2$ (for fixed $[\boldsymbol{a}]$) allow to reconstruct $\boldsymbol{w}, \boldsymbol{r}$. But Protocol Σ_{std} is not particularly communication-efficient, as it sends the full masked witness $\boldsymbol{z} \in \mathbb{F}_p^n$ as well as $[\boldsymbol{a}] \in \mathbb{G}^m$. Using probabilistic verification, one can often improve this.

4.1.1.1. Probabilistic Verification

The underpinning of *efficient* arguments of knowledge (without zero-knowledge) is probabilistic verification of the claim. For instance, instead of verifying $[\boldsymbol{A}]\boldsymbol{w} = [\boldsymbol{t}]$ directly, the verifier could send a random $y \xleftarrow{\$} \mathbb{F}_p$. Both parties compute $\boldsymbol{y} = (y^i)_i \in \mathbb{F}_p^m$ and prove (resp. verify) $[\widehat{\boldsymbol{A}}]\boldsymbol{w} = [\widehat{t}]$ for $[\widehat{\boldsymbol{A}}] = \boldsymbol{y}^\top[\boldsymbol{A}] \in \mathbb{G}^{1 \times n}$ and $[\widehat{t}] = \boldsymbol{y}^\top[\boldsymbol{t}] \in \mathbb{G}$ instead. This would result in a communication complexity which is independent of m as $[\widehat{a}] = [\widehat{\boldsymbol{A}}]\boldsymbol{r} \in \mathbb{G}$.

Not all probabilistic verifications are alike. To work well with zero-knowledge, we need "suitable" verification procedures, so that techniques to efficiently attain zero-knowledge are applicable. This essentially means that the verification should be *linear*, i.e. all tested equations should be linear. (Abstract groups only allow linear operations anyway.)

We define so-called *testing distributions* which are distributions over $\mathbb{F}_p^n$, yielding "random linear test maps". Given enough independent test maps and images, one can recover the "tested object". This allows to extract knowledge. Our definitions are tailored to our setting.[1]

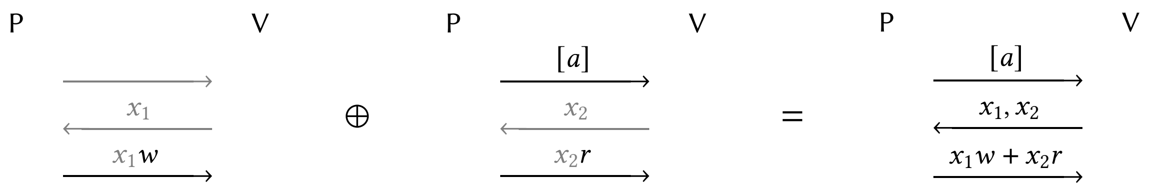


Figure 4.1.: Linear Combination of Protocols. *Left:* The trivial proof of knowledge: Send the witness. *Middle:* Send a random statement. Then send the witness. *Grayed out:* Terms for linear combination. *Right:* The linear combination with verifier's randomness.

[1] See [Wik18] for a possible generalisation, which considers "metroids".

4.1.1.2. Linear Combinations of Protocols

A core insight for achieving zero-knowledge (and reducing communication) in our setting *efficiently* is that protocols can often be linearly combined, see Fig. 4.1 for an illustration. This exploits the *linearity* of the computations and checks of verifier and prover in each round. By running an "umasked *non*-zero-knowledge argument" (Fig. 4.1, left) and linearly combining it with an argument for a "masking randomness" (middle), one can achieve zero-knowledge (right). All of our zero-knowledge compilations rely on this strategy. We typically consider *random* linear combinations of protocols, where the verifier picks the randomness (x_1, x_2 in Fig. 4.1), as this often achieves extractability. In fact, this kind of linear combination recovers the batch proofs of [PBD07], see Section 4.3.4. Nevertheless, non-randomised linear combinations are also useful, e.g. in Protocols 4.4.1 and B.4.1 or [BBB+18] they are used to compress multiple commitments into one.

4.1.1.3. Uniform(-or-Unique) Responses

In our setting, for simulation it is typically enough to ensure that the prover's messages are distributed uniformly at random. More concretely, the responses should be either uniformly distributed (conditioned on all *later* messages, *not* previous messages), such as $\boldsymbol{z}$ in Protocol Σ_{std}. Or they should be uniquely determined and *efficiently* computable from the challenges and all *later* messages, such as $[\boldsymbol{a}]$ in Protocol Σ_{std}. This allows to construct a trivial simulator, which constructs the transcript *in reverse*: Starting with the final messages, and working its way towards the beginning, the simulator picks the uniformly distributed messages itself, and then computes the uniquely determined ones. All simulators in this chapter work like this.

4.1.1.4. Kernels and redundancy

Many interesting statements are non-linear. For example, for polynomial commitments [BG18], we want to show that $[c] \in \mathbb{G}$ is a commitment to a polynomial $f \in \mathbb{F}_p[X]$ (of degree at most $d-1$) and $f(x) = t$, where $x \in \mathbb{F}_p$ is a random challenge. Naively, one commits to the coefficients of the polynomial with monomial basis X^i for $i = 0, \ldots, n-1$. Suppose we have a (linear) protocol which proves $f(x) = t$. We could hope that running a random linear combination as in Fig. 4.1 should give us uniform-or-unique responses (and hence zero-knowledge). However, we are in a predicament: For random $g \in \mathbb{F}_p[X]$, we have $(f+g)(x) \neq f(x)$ and thus we have to let V know $y = g(x)$ somehow. To ensure the prover does not send arbitrary y, we have to rely on a proof again! But if this proof leaks (too much) information,[2] we cannot use it to randomise the response. We can escape this cycle by having a way to randomize *without changing the statement.* In other words, we need some g with $g(x) = 0$ for all $x \in \mathbb{F}_p$. Clearly, that means $g = 0$, and there's nothing random anymore.

One solution is to add *redundancy*, which does not "influence" soundness: Here, we artificially create a non-trivial kernel of the "evaluate at x"-map. We can do so by representing $f(X)$ as $\sum_i (\alpha_i + \beta_i) X^i$ and commit to all α_i and β_i. Now we can mask with $g(X)$ where $\alpha_i \xleftarrow{\$} \mathbb{F}_p$ and $\beta_i = -\alpha_i$. Thus, we successfully injected randomness into the response. Generally, adding just enough redundancy to achieve uniformly random responses is our goal.

[2] If it only leaks a little bit, then a linear combination with many g_i may work.

4.1.1.5. Composition of arguments systems

For completeness, we recall that, by committing to (intermediate) results and sharing these commitments in multiple argument systems, one can easily combine the most efficient arguments for each task. While this is an obvious approach, it is easy to overlook as NP-complete languages offer a suitably encoded statement as a straightforward alternative.

4.1.1.6. Exemplary applications of the basic techniques

As in [BBB+18], we construct $\mathsf{IPA}_{\mathrm{noZK}}$ from a linear combination of the logarithmic communication linear map preimage arguments ($\mathsf{LMPA}_{\mathrm{noZK}}$) for $\exists \boldsymbol{w}\colon [\boldsymbol{g}]\boldsymbol{w} = [\boldsymbol{t}]$. The linear combination compresses the messages from multiple instances into a single one. For our logarithmic communication (almost) zero-knowledge inner product argument $\mathsf{IPA}_{\mathrm{almZK}}$ for $\exists \boldsymbol{x}, \boldsymbol{y}\colon \langle \boldsymbol{x}, \boldsymbol{y}\rangle = t$, we mask the witness as $\langle \boldsymbol{x}+\boldsymbol{r}, \boldsymbol{y}+\boldsymbol{s}\rangle = t$, with masks $\boldsymbol{r}, \boldsymbol{s}$ chosen such that $\langle \boldsymbol{r}, \boldsymbol{y}\rangle = \langle \boldsymbol{r}, \boldsymbol{s}\rangle = \langle \boldsymbol{x}, \boldsymbol{s}\rangle = 0$. This is an application of the “redundancy/kernel” technique. The “uniform-or-unique” guideline ensures that it is enough that each response is random. By choosing the random components in $\boldsymbol{r}, \boldsymbol{s}$ suitably, a logarithmic number of randomized components suffice to achieve uniform responses (and all other components may be 0). Finally, for our logarithmic communication shuffle argument Π_{shuffle} (Appendix B.2), we compose $\mathsf{QESA}_{\mathrm{ZK}}$ (our quadratic equation argument) and $\mathsf{LMPA}_{\mathrm{ZK}}$ by sharing a commitment to the witness.

4.1.2. Contribution

Our contribution is two-fold. On the one hand, we extract and present useful strategies and building blocks from [BCC+16; BBB+18] and devise new protocols with improved performance (and a more expressive language). On the other hand, we initiate the study of more precise soundness notions to assert better provable security guarantees.

4.1.2.1. New Protocols

As a minor contribution, we note that there is no work which outlines the — certainly folklore — techniques from Section 4.1.1, in particular linear combination of protocols, as useful guidelines for efficient zero-knowledge protocols. Implicitly, these techniques are used in many works, e.g. [PBD07; Gro09; BCC+16; BBB+18; BG18] to cite a few. We follow the above guidelines for constructing and explaining our zero-knowledge arguments.

Linear Map Preimage Argument (LMPA) We give, in two steps, an argument for $\exists \boldsymbol{w}\colon [\boldsymbol{A}]\boldsymbol{w} = [\boldsymbol{t}]$ for $[\boldsymbol{A}] \in \mathbb{G}^{m\times n}$ with communication $\mathcal{O}(\log(n))$. The idea is to first use batch verification. Essentially, $\mathsf{LMPA}_{\mathrm{batch}}$ multiplies the equation with a random vector $\boldsymbol{y} \in \mathbb{F}_p^m$ from the left to obtain $[\widehat{\boldsymbol{A}}] = \boldsymbol{y}^\top[\boldsymbol{A}] \in \mathbb{G}^{1\times n}$ and $[\widehat{t}] = \boldsymbol{y}^\top[\boldsymbol{t}] \in \mathbb{G}$. Thus, communication is independent of m. Now, we prove $\exists \boldsymbol{w}\colon [\widehat{\boldsymbol{A}}]\boldsymbol{w} = [\widehat{t}]$ using the logarithmic-communication argument $\mathsf{LMPA}_{\mathrm{noZK}}$, which is derived from [BCC+16] $\mathsf{LMPA}_{\mathrm{noZK}}$ can be made zero-knowledge by fully masking the witness, which we denote by $\mathsf{LMPA}_{\mathrm{simpleZK}}$. Thus, the total communication cost (resp. computation cost) of $\mathsf{LMPA}_{\mathrm{simpleZK}}$ in terms of group elements (resp. group operations) becomes $\mathcal{O}(m\log(n))$ (resp. $\mathcal{O}(mn)$). By first using to $\mathsf{LMPA}_{\mathrm{batch}}$ and then $\mathsf{LMPA}_{\mathrm{simpleZK}}$, the cost is $\mathcal{O}(\log(n))$ resp. $\mathcal{O}(mn)$, as claimed.

For completeness, we also derive a more sophisticated approach, where the actual (additive) overhead in computation for achieving zero-knowledge is much smaller (namely, logarithmic in n). However, due to the incurred complexity and subtleties, $\mathsf{LMPA}_{\text{simpleZK}}$ (with linear overhead) is the preferable choice in practice.

Quadratic Equation Commit-and-Prove First of all, we derive a (almost) zero-knowledge inner product argument $\mathsf{IPA}_{\text{almZK}}$ from [BCC+16; BBB+18], with constant communication and logarithmic computational overhead compared to [BCC+16; BBB+18]. Based on $\mathsf{IPA}_{\text{almZK}}$, we construct an argument for proving $\exists \boldsymbol{w}\colon \forall i\colon \langle \boldsymbol{w}, \boldsymbol{\Gamma}_i \boldsymbol{w}\rangle = 0$, where $\boldsymbol{\Gamma}_i \in \mathbb{F}_p^{n\times n}$ are public matrices and $\boldsymbol{w}$ is a (committed) vector. For efficiency, we carry out a batch proof, i.e. we prove $\langle \boldsymbol{w}, \boldsymbol{\Gamma}\boldsymbol{w}\rangle$ with $\boldsymbol{\Gamma} := \sum_i r_i \boldsymbol{\Gamma}_i$ for random $r_i \in \mathbb{F}_p$. The resulting argument, $\mathrm{QESA}_{\mathrm{ZK}}$ for short, is “adaptive commit-and-prove”, i.e. the statement $\boldsymbol{\Gamma}_i$ may be chosen after the commitment to $\boldsymbol{w}$.

The commit-and-prove system $\mathrm{QESA}_{\mathrm{ZK}}$ is conceptually simple. We extend $\mathrm{QESA}_{\mathrm{ZK}}$ to $\mathrm{QESA}_{\mathrm{Copy}}$ to demonstrate that our techniques can be easily extended and combined with other argument systems.

Quadratic Equations and R1CS Being able to prove arbitrary quadratic equations instead of R1CS equations, i.e. equations of the form $(\sum a_i x_i)(\sum b_i x_i) + \sum c_i x_i = 0$, gives much flexibility. To the best of our knowledge, expressing the quadratic equation $\langle \boldsymbol{x}, \boldsymbol{x}\rangle = \sum x_i^2 = t$ as R1CS requires n equations: $y_i = x_i^2$ $(i = 1, \dots, n-1)$ and $x_n^2 = t - \sum_i y_i$, where y_i are auxiliary variables. Requiring n equations is surprising for [BCC+16; BBB+18] which build on an *inner product argument.* Clearly, $\mathrm{QESA}_{\mathrm{ZK}}$ needs one (quadratic) equation to express $\langle \boldsymbol{x}, \boldsymbol{x}\rangle = t$. We discuss the relation between R1CS and QE in terms of them being the “native language of a proof system” in more detail in Appendix B.8.

Example 4.1.1 (QE for polynomial evaluation). Using general quadratic equations, one can evaluate any (univariate) polynomial $f(X) = \sum_{i=0}^{d^2-1} a_i X^i$ of degree $d^2 - 1$ with $2d$ equations and intermediate variables. Concretely, let $y_i = x^i = y_{i-1}x$, $z_i = x^{di} = z_1 z_{i-1}$, for $i = 2, \dots d-1$ and $z_1 = y_{d-1}x$ and $z_0 = 1$. Then $f(x) = \sum_{i,j=0}^{d} a_{i+jd} y_i z_j$. Using this, one can speed up “table lookups”, which are typically encoded as polynomial evaluation. Note however, that by using composition of protocols, even more efficient (batch) subproofs for such tasks may be possible.

Example 4.1.2. For S(N)ARK-friendly cryptography [KZM+15], supporting quadratic equations is very useful. Matrix-vector multiplications are efficient even when both matrix and vector are *secret.* “Embedding” an elliptic curve (see Jubjub [Web]), is also more efficient than for R1CS. For general point addition in a (twisted) Edwards curve, we need 5 instead of 7 constraints per bit.

Example 4.1.3. The square decompositions in the relaxed range proof from Chapter 3 are naturally quadratic equations, and can be proven efficiently combined with our constructions, leading to an alternative range proof which required asymptotically fewer constraint for (very) large batch sizes, cf. Example 3.5.2 in Section 3.5.4.

Correctness of a Shuffle By instantiating the shuffle proof of Bayer and Groth [BG12] with $\mathsf{LMPA}_{\mathrm{ZK}}$ and $\mathrm{QESA}_{\mathrm{ZK}}$ as subprotocols, we obtain an argument Π_{shuffle} for correctness of a shuffle (of ElGamal ciphertexts). To the best of our knowledge, this is the first such (fully specified) efficient argument with proof size $\mathcal{O}(\log(N))$.[3] Our computational efficiency is comparable to [BG12], which has proof size $\mathcal{O}(\sqrt{N})$. More concretely, we estimate the relative overhead at about 2–3×.

4.1.2.2. Knowledge Errors, Tightness and Short-Circuit Extraction

As an intermediate abstraction, we define the notion of testing distributions. This allows us to view the verifier's challenge distributions as a separate object, which may be modified to achieve different trade-offs in a scheme. In particular, it is possible to choose testing distribution which lead to different, tunable levels of soundness, e.g. a knowledge error of 2^{-120} instead of 2^{-256}, which impacts runtime positively.

Short-Circuit Extraction. To motivate a more precise analysis of extraction, consider the extractor in [BBB+18]. According to [JT20], it needs $7(L+1)M^3$ transcripts, for a statement consisting of L linear and M multiplicative constraints. If we assume $L \approx M \approx n$, where n is the number of variables, we derive a runtime induced security loss of $\mathcal{O}(n^4)$ and concretely $\approx 2^{80}$ for $n \approx 2^{20}$.

To improve upon this, we define the notion of *short-circuit extraction*, which is applicable to extraction assertions of the form "Ext either finds a witness *or* it solves a hard problem". It formalises the (common) behaviour of an extractor to either find a witness with *few* transcripts, or solve the hard problem (e.g. equivocating a commitment). Without distinguishing these cases, the bounds on the necessary number of transcripts for extraction is much higher. For example, we show that the extractor for the $\mathsf{LMPA}_{\mathsf{ZK}}$ and $\mathsf{IPA}_{\mathsf{almZK}}$ (and also [BCC+16; BBB+18]) needs to explore a tree of transcripts of size $\mathcal{O}(n)$ in the worst case. This improves the runtime induced security loss from $\approx n^2$ to $\approx n$ for the inner product argument.

For $\mathsf{QESA}_{\mathsf{ZK}}$, extracting a proof for N quadratic equations in n variables requires $\mathcal{O}(nN)$ transcripts, which for $n, N \approx 2^{20}$ implies a runtime induced security loss of $\approx 2^{40}$. By using a special testing distribution, this can be further reduced to $\mathcal{O}(\log(nN)n)$, and $\approx 2^{30}$ for $n, N \approx 2^{20}$ Thus, we reduce the runtime induced quartic loss $\mathcal{O}(n^4)$ in n to a merely quasi-linear loss $\mathcal{O}(n\log(n))$.

In Appendix B.5.1, we lay out an intuitive connection between communication efficiency and extraction efficiency, which implies that extraction from $\mathcal{O}(\frac{n}{\log(n)})$ transcripts would be optimal (under strong assumptions). As a consequence, the special instantiation of $\mathsf{QESA}_{\mathsf{ZK}}$ (Corollary 4.4.12) is close to optimal, as it requires $\mathcal{O}(n\log(n))$ transcripts for extraction (assuming $n \approx N$ in the asymptotics). We also elaborate on a loophole in above security estimates, namely how to *efficiently obtain* the transcripts. For this, we present candidate algorithms in Appendix B.9 whose (optimal) runtime bounds are only conjectural or heuristically derived.

Dual Testing Distributions Dual testing distributions are a technical tool which allow us to sample a "new" commitment key from a given one, such that knowledge (e.g. commitment opening) cannot be transferred. This turns out to be more communication efficient than letting the verifier send a new commitment key. To the best of our knowledge, this is a new technique.

[3] In [BBB+18], a (less efficient) shuffle proof is claimed, which apparently handles Pedersen commitments (not ElGamal ciphertexts). After publication of this work [HKR19b], of an implementation of shuffle proofs based on of Bulletproofs [AVY] with a similar approach as ours, but which also seems to operate on commitments, not ciphertexts. See Remark B.2.1 for further discussion.

4.1.2.3. Efficiency and Comparison to Bulletproofs [BBB+18]

In Table 4.1, we compare our argument systems with related work in the group setting. In Table 4.2, we give precise efficiency measures for $\mathsf{LMPA}_{\mathsf{ZK}}$ and $\mathsf{QESA}_{\mathsf{ZK}}$. In any case, $n = |\boldsymbol{w}|$ is the size of the witness $\boldsymbol{w} \in \mathbb{F}_p^n$. Since it is statement dependent, we ignore that QE is more powerful than R1CS, possibly allowing smaller witness size (as seen in the example $\langle \boldsymbol{x}, \boldsymbol{x} \rangle = t$ above). Since statement size N is typically a small mulitple of witness size, we ignore its influence. In Table 4.2, we omit the verifier's computation, since after optimisations [BBB+18], both are almost identical.[4] Generally, optimisations applicable to [BBB+18] are applicable to our protocols as well. For the prover, we do not optimise (e.g. we use no multi-exponentiations), and are not aware of non-generic optimisation. Although $\mathsf{QESA}_{\mathsf{ZK}}$ and $\mathsf{QESA}_{\mathsf{Copy}}$ cover general quadratic equations, they compare favorably to Bulletproofs [BBB+18] which only cover R1CS. By default, they prove slightly different statements than Bulletproofs, see Remark 4.4.18, hence the parameters are not perfectly comparable. In Section 4.5, we compare our implementations of (aggregate) range proofs. Due to their close relation, we also compare with Bulletproofs+ [CHJ+22], which were published after this work (which appeared as [HKR19a]).

	URS	Ass.	Moves	Comm.	Comp. P	Comp. V	Nat. $\mathcal{R}$
SNARG [GGPR13]	✗	KoE	1	$\mathcal{O}(1)$	$\mathcal{O}(n)$	$\ll \lvert\boldsymbol{w}\rvert$	R1CS
Bulletproofs[BBB+18]	✓	dlog	$\mathcal{O}(\log(n))$	$\mathcal{O}(\log(n))$	$\mathcal{O}(n)$	$\lvert\boldsymbol{w}\rvert$	R1CS
This work	✓	dlog	$\mathcal{O}(\log(n))$	$\mathcal{O}(\log(n))$	$\mathcal{O}(n)$	$\lvert\boldsymbol{w}\rvert$	QE

Table 4.1.: **URS:** Is a common uniformly *random* string (URS) sufficient for setup, a.k.a. transparent setup? **Ass(umption):** Underlying security assumption. Knowledge of exponents (KoE); Hardness of dlogs. **Moves:** The number of messages sent. **Comm(unication):** The number of group elements sent. **Comp:** Computation of P resp. V in number of exponentiations. **Nat(ive)** $\mathcal{R}$**:** "Native" relation proven.

	Comm. $\mathbb{G}$	Comm. $\mathbb{F}_p$	Comp. P	$\mathcal{R}$
$\mathsf{LMPA}_{\mathrm{simpleZK}}$	$\lessapprox 2km\log_k(n)$	k	$\lessapprox 2mn$	LMP
$\mathsf{LMPA}_{\mathrm{batch+simpleZK}}$	$\lessapprox 4k\log_k(n)$	k	$\lessapprox (m+5)n$	LMP
Bulletproofs[BBB+18]	$2\lceil\log(n)\rceil + 8$	5	$\lessapprox 12n$	R1CS
Bulletproofs+[CHJ+22]	$2\lceil\log(n)\rceil + 5$	3	$\lessapprox 11n$	R1CS
$\mathsf{QESA}_{\mathsf{ZK}}/\mathsf{QESA}_{\mathsf{Copy}}(k=2)$	$2\lceil\log(n+3)\rceil + 3$	2	$\lessapprox 8n$	QE

Table 4.2.: Detailed comparisons in terms of group operations. By "$\lessapprox$" we denote upper bounds up to logarithmic (or constant) additive terms, i.e. $g \lessapprox f$ means $g \leq f + \mathcal{O}(\log(f))$. Note that k is a tunable parameter but $k = 2$ is the sweet spot.[5]We assume all random exponents are full sized and do not count multi-exponentiations. For $\mathsf{QESA}_{\mathsf{Copy}}$, we assume only Pedersen commitments to a single value are given as inputs (matching [BBB+18]), see Remark 4.4.18 for minor inaccuracies and differences due to this comparison.

4.1.2.4. Comparison with Other Proof Systems

It is hard to make a fair comparison of proof systems. There are many relevant parameters, such as setup, assumptions, quantum resistance, native languages, etc., beyond mere proof size and performance. See Section 4.1.3 for a high-level discussion. To draw sensible conclusions from comparisons on an implementation level, one should compare fully optimised implementations. Thus, we restrict ourselves to a comparison with Bulletproofs (which we reimplemented with the same optimisation level as our

[4] For completeness, this means $4n$ resp. $6n$ exponentiations for $\mathsf{QESA}_{\mathsf{ZK}}$ resp. [BBB+18]. The verifier in $\mathsf{LMPA}_{\mathsf{ZK}}$ needs $\approx mn$ group exponentiations.

[5] For $k > 2$, FFT-based techniques are needed to ensure the prover's complexity, which in fact improves for larger k. See the discussion in [BCC+16], which adapts to our setting.

proof systems). For concrete numbers regarding (implementation) performance, as well as other factors relevant to the comparison of proof systems, we refer to [BSCR+18, Figure 2]. Our proof systems are similar enough to Bulletproofs for these comparisons to still hold.

4.1.2.5. Implementation

In Section 4.5, we compare our implementations of (aggregate) range proofs. The theoretical prediction of 0.66–0.75× prover runtime compared to [BBB+18] is close to measurements, which suggest 0.7×. Using 140bit challenges, we experimentally attain $\approx 0.63\times$ compared to [BBB+18] on the same platform. We stress that, we compare the dedicated range proofs of [BBB+18] with our generic instantiation of $\mathsf{QESA}_{\mathsf{ZK}}$, since this chapter does not focus on range proofs.

4.1.3. Related Work

Due to space constraints, we only elaborate on the most important concepts and related (mostly subsequent) works. We refer to [ZkpComref] for an overview of the current landscape and a general taxonomy of zero-knowledge proof systems.

The DLOG Setting and ILC. Very closely related works are [Gro09; BCC+16; BBB+18; BG18; BAZB20; AC20; CHJ+22], which are efficient proofs in the dlog setting. Subsequent works [BAZB20; CHJ+22; AC20] also build one the folding technique of Bulletproofs [BCC+16; BBB+18], enhancing them differently. For example, the work [BAZB20] also provides better interoperability with other Σ-protocols. Other variations of Bulletproofs or proofs based on the folding technique include the work of Lai, Malavolta, and Ronge [LMR19] which extends Bulletproofs to bilinear groups and statements, and the "compressed Σ-protocol theory" [AC20; ACK21; ACR21], which uses folding and linear map preimage argument (instead of an IPA)[6] as its privotal compression technique. Bootle, Chiesa, and Sotiraki [BCS21] show that the sumcheck paradigm can explain the folding technique.

Moreover, many zero-knowledge proofs in the group setting are instantiations of [CD98; Mau15]. The possibilities of our setting, namely ability to apply linear transformations to a committed witness has been abstracted in the ideal linear commitment model [BCG+17]. (Our techniques for $\mathsf{QESA}_{\mathsf{ZK}}$ are amenable to ILC.)

Zero-Knowledge (Weighted) Inner Product Arguments of [CHJ+22]. This work also build upon a zero-knowledge IPA and hence very closely to our approach. One core difference in the approach of [CHJ+22] and this work, is that [CHJ+22] considers a specific (Pedersen) commitment scheme with designated randomness terms, and it refreshes the randomness in each round. This simplifies achieving zero-knowledge. Our approaches tried to avoid designated randomness terms, though in hindsight, they appeared in some places anyway (e.g. in commitment randomness in $\mathsf{QESA}_{\mathsf{ZK}}$). Moreover, we use an up-front masking instead of injecting fresh randomness in each round. It should be possible to use the zero-knowledge (weighted) inner product argument (zk-WIP) of [CHJ+22] in place of our "almost" zero-knowledge $\mathsf{IPA}_{\mathsf{almZK}}$ within $\mathsf{QESA}_{\mathsf{ZK}}$ and variants. This does not affect the proof size, but achieves perfect HVZK (instead of statistical) and may allow to replace the terms $\log_k(n+3)$ by $\log_k(n)$ in the proof size.

[6] Except for [ACR21], which generalizes [LMR19] and still needs to partially rely on an IPA for logarithmic communication.

Sublinear Verifiation in the DLOG Setting, Knowledge Assumptions and the SRS Setting. Many works (in the DLOG setting) which rely on “naive” folding and uniformly random strings (URS) instead of structured reference strings (SRS) as their common reference string (CRS) have sublinear communication but a *quasi-linear-time* verifier, e.g. [BCC+16; BBB+18; BAZB20; AC20; CHJ+22]. Much effort has gone into improving the situation, and there are now practically efficient arguments with sublinear communication and sublinear verification with a URS. Interestingly, we do not know group-based examples which rely solely on DLOG, but all use either bilinear groups or groups of hidden order, e.g. [BFS20; Set20; BDFG21; Lee21]. However, arguments based on SRSs (in bilinear groups or groups of hidden order), are still more efficient, although many additionally rely on knowledge assumptions, e.g. the line of works[GGPR13; DFGK14; Gro16] even attain constant size proofs. As such, there is now a body of work providing a theoretical treatment [KMSV21] of and protocols [BGM17] for so-called “setup ceremonies” which securely compute a SRS. Moreover, a middle ground between URSs and SRSs, namely *updatable* SRSs, has been conceived and is being explored [GKM+18; MBKM19; GWC19; CHM+20; RZ21; CFF+21].

Lattices, PCPs, IOPs, MPC-in-the-Head, and More. Efficient lattice-based argument system have advanced significantly, and the work [BLNS20] is essentially a translation of the folding approach to the lattice setting. Techniques, such as probabilistically checkable proofs (PCP), MPC-in-the-head [IKOS07], interactive oracle proofs (IOP) [BCS16; RRR16] and more, construct efficient zero-knowledge proofs without relying on public key primitives. The possible performance gain (and quantum resistance) is interesting from a practical point of view. As there is too much work of interest to adequately cover here, we refer again to [ZkpComref] for further references.

Tightness of the Security Reduction. Jaeger and Tessaro [JT20] provide a tighter analysis of the extractor in [BCC+16], improving the knowledge soundness. In [ACK21] a modified extractor is shown to be essentially optimal for special sound protocols (both in knowledge soundness and runtime). Neither of the works [JT20; ACK21] exploit short-circuit extraction, resulting in quartic runtime tightness for Bulletproofs and $\mathsf{QESA}_{\mathsf{ZK}}$ (in the witness size) for general constraints, whereas short-circuit extraction suggest that linear runtime tightness is possible for $\mathsf{QESA}_{\mathsf{ZK}}$. Another line of works [GT21; GOP+22] analyzes the security of Bulletproofs (with Fiat–Shamir transformation applied) using the algebraic group model [FKL18], which is a (strong) knowledge assumption. Perhaps (not) surprisingly, for algebraic adversaries, they achieve essentially optimal tightness results (both in knowledge error and runtime) since their extraction is straight-line, i.e. does not rewind the adversary. Note that this is not a contradiction to our lower bound on runtime tightness, since it only holds for *black-box* extractors, and the AGM is inherently non-black-box.

4.1.4. Structure of this Chapter

In Section 4.2, we clarify additional preliminaries and notational conventions, in particular, testing distributions and short-circuit extraction. In Section 4.3, we analyze the folding technique for linear map preimage arguments. We apply this in Section 4.4 to construct our zero-knowledge inner product argument and our quadratic equation satisfiability argument. Finally, in Section 4.5, we provide an overview of our implementation and benchmarks.

4.2. Preliminaries

The number $p \in \mathbb{N}$ will always denote a prime, $\mathbb{F}_p := \mathbb{Z}/p\mathbb{Z}$, and $\mathbb{G}$ is a (cyclic abelian) group of order p. We use additive implicit notation for $\mathbb{G}$ as introduced in [EHK+13]. Recalling Section 2.2.1, we write $[1]$ for some (fixed public) generator associated with $\mathbb{G}$ and $[x] := x[1]$. We extend this notation to vectors and matrices, i.e. for compatible $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$ over $\mathbb{F}_p$, we write $\boldsymbol{A}[\boldsymbol{B}]\boldsymbol{C} = [\boldsymbol{ABC}]$. Matrices are bold, e.g. $[\boldsymbol{a}]$, components not, e.g. $[a_i]$. By $\boldsymbol{e}_i$ we denote the i-th standard basis vector. We write $\operatorname{diag}(\boldsymbol{M}_1, \dots, \boldsymbol{M}_n)$ for a block-diagonal matrix. By id_n we denote the $n \times n$ identity matrix.

Matrices of matrices. As a special form of block matrices, we use matrices of matrices and vectors of vectors. Matrices of matrices are of the form $R^{m\times n}$, where instead of $R = \mathbb{F}_p$ we have $R = \mathbb{F}_p^{\mu\times\nu}$. For example, a matrix of matrices $\boldsymbol{M} = \begin{pmatrix} \boldsymbol{M}_{1,1} & \boldsymbol{M}_{1,2} \\ \boldsymbol{M}_{2,1} & \boldsymbol{M}_{2,2} \end{pmatrix}$. There is an evident bijection between such block matrices in $(\mathbb{F}_p^{\mu\times\nu})^{m\times n}$ and ordinary matrices in $\mathbb{F}_p^{\mu m\times\nu n}$ via forgetting or adding the blocks. The general multiplication of two matrices of matrices is thus merely (block) matrix multiplication. That is, for $\boldsymbol{A} \in S^{\ell\times m}$ where $S \in \mathbb{F}_p^{\lambda\times\mu}$ and $\boldsymbol{M}$ is as above, we define $\boldsymbol{Z} = \boldsymbol{AM} \in (\mathbb{F}_p^{\lambda\times\nu})^{\ell\times n}$ via $\boldsymbol{Z}_{i,j} = \sum_{\ell=1}^m \boldsymbol{A}_{i,\ell}\boldsymbol{M}_{\ell,j}$. When transposing matrices of matrices, we are explicit about whether or not the inner matrices are transposed or not, i.e. whether we consider $(\boldsymbol{M}^\top)_{i,j} := \boldsymbol{M}_{j,i}$ or $(\boldsymbol{M}^\top)_{i,j} := \boldsymbol{M}_{j,i}^\top$, e.g. by specifying the dimension of $\boldsymbol{M}^\top$.

We also use following special case: Matrix multiplication of a matrix $\boldsymbol{M} \in R^{m\times n}$ with a matrix $\boldsymbol{A}$ in $\mathbb{F}_p^{\ell\times m}$ is defined as $\boldsymbol{Z} = \boldsymbol{AM} \in R^{\ell\times n}$, where $\boldsymbol{Z}_{i,j} = \sum_{\ell=1}^m A_{i,\ell}\boldsymbol{M}_{\ell,j}$. Effectively, this interprets $\boldsymbol{A} \in \mathbb{F}_p^{\ell\times m}$ as a block-diagonal matrix $(\boldsymbol{A}'_{i,j})_{i,j}$ in $S^{\ell\times m}$ and $S = \mathbb{F}_p^{\mu\times\mu}$ with $\boldsymbol{A}'_{i,j} = A_{i,j} \cdot \mathrm{id}_\mu$, that is, as $\boldsymbol{A}' = \mathrm{id}_\mu \otimes \boldsymbol{A}$ where "$\otimes$" is the tensor/Kronecker product. Note that whenever multiple definitions apply (e.g. if R or S is $\mathbb{F}_p$) all yield identical results.

4.2.1. Matrix Kernel Assumptions and Pedersen Commitments

Instead of discrete logarithm assumptions, the generalisation of hard (matrix) kernel assumptions [MRV16], but for right-kernels, better suits our needs.

Definition 4.2.1. Let $\mathbb{G} \xleftarrow{\$} \mathsf{GrpGen}(1^\lambda)$ be a group generator (we let $[1]$ and p be implicitly given by $\mathbb{G}$). Let $\mathcal{D}_{m,n}$ be a (efficiently samplable) distribution over $\mathbb{G}^{m\times n}$ (where m and n may depend on λ). We say $\mathcal{D}_{m,n}$ has a **hard kernel assumption** if for all efficient adversaries $\mathcal{A}$, we have

$$\Pr\Big[\mathbb{G} \xleftarrow{\$} \mathsf{GrpGen}(1^\lambda); [\boldsymbol{A}] \xleftarrow{\$} \mathcal{D}_{m,n}; \boldsymbol{x} \xleftarrow{\$} \mathcal{A}(1^\lambda, \mathbb{G}, [\boldsymbol{A}]) \colon\ [\boldsymbol{A}]\boldsymbol{x} = 0 \ \wedge\ \boldsymbol{x} \neq 0\Big] \leq \mathsf{negl}(\lambda)$$

For simplicity, we will often only implicitly refer to $\mathcal{D}_{m,n}$ and just say $[\boldsymbol{A}]$ has hard kernel assumption. Note that kernel assumptions generalise discrete log assumptions: Finding a non-trivial kernel element of $[h, 1] \in \mathbb{G}^2$ immediately yields the discrete logarithm h of $[h]$.

If $\mathcal{D}_{m,n}$ is a matrix distribution with hard kernel assumption, then $[\boldsymbol{A}] \xleftarrow{\$} \mathcal{D}_{m,n}$ is a (Pedersen) commitment key ck. Commit to $\boldsymbol{x} \in \mathbb{F}_p^n$ via $\mathsf{Com}_{ck}(\boldsymbol{x}) = [\boldsymbol{c}] \in \mathbb{G}^m$. Breaking the binding property of the commitment is equivalent to finding non-trivial elements in $\ker([\boldsymbol{A}])$. The common case will be $[\boldsymbol{g}] \in \mathbb{G}^{1\times(n+1)}$ drawn uniformly as commitment key ck. Breaking the hard kernel assumption for $[\boldsymbol{g}]$ is tightly equivalent to breaking the dlog assumption in $\mathbb{G}$. Write $\boldsymbol{x} = (r_{\boldsymbol{w}}, \boldsymbol{w})$ with $r_{\boldsymbol{w}} \in \mathbb{F}_p$, $\boldsymbol{w} \in \mathbb{F}_p^n$. If $r_{\boldsymbol{w}} \xleftarrow{\$} \mathbb{F}_p$ is drawn uniformly, it is evident that $[c] = [\boldsymbol{g}]\boldsymbol{x}$ perfectly hides $\boldsymbol{w}$, i.e. $[c]$ is uniformly distributed in $\mathbb{G}$.

Remark 4.2.2. It would be convenient to consider *hard kernel assumptions with prior knowledge $V \leq \mathbb{F}_p^n$*, where the subvector space V is some (leaked) knowledge about $\ker([\boldsymbol{B}])$ for $[\boldsymbol{B}] \xleftarrow{\$} \mathcal{D}_{m,n}$. The reason being that we construct matrices $[\boldsymbol{B}]$ from $[\boldsymbol{A}]$ (where $[\boldsymbol{A}]$ has a standard hard kernel assumption) in such a way that some kernel elements are known, e.g. because $[\boldsymbol{B}]$ has some zero columns. However, to keep the overhead low, we deal with these cases explicitly.

Remark 4.2.3 (Efficient group operations). In general, multiplications of group elements with small scalars ("small exponents") are faster, since computational complexity for a scalar-group multiplication is roughly linear in the bit-size of the scalar. Thus, it is beneficial for efficiency to keep scalars small. In certain situations, e.g. terms of the form $\sum_{i=1}^{n} \alpha_i [h_i]$ for $\alpha_i \in \mathbb{F}_p$, $[h_i] \in \mathbb{G}$ (so-called multi-exponentiations), further optimizations exist, see e.g. [MOV96]. Moreover, if the α_i are structured, special-purpose optimizations exist. Concretely, if $\alpha_i = \xi^{i-1}$ for $\xi \in \mathbb{F}_p$, then $[a] = \sum_{i=1}^{n} \alpha_i [h_i]$ may be computed by setting $[a_n] = [h_n]$ and then letting $[a_i] = \xi \cdot [a_{i+1}] + [h_i]$ for $i = n-1, \dots, 1$, where $[a_1] = [a]$ (i.e. by using Horner's scheme for polynomial evaluation). In particular, if ξ is small, then $[a]$ is computed with "small exponentiations" only (even though most α_i are fully exponents). This is special case appears frequently in our protocols.

4.2.2. Testing Distributions

Intuitively, testing distributions are a special form of probabilistic linear verification where one can *efficiently* recover the "tested" value given enough "tests". Thus, they are used to recover the witness in proofs of knowledge. We only define testing distributions over $\mathbb{F}_p^m$. Moreover, we only use them as an abstraction which allows to consider small variations of protocols, obtained by changing the testing distribution. As we will see, the choice testing distribution can affect efficiency optimizations, and also affects what security guarantees we are able to prove.

Example 4.2.4. To test if a vector $[\boldsymbol{c}] \in \mathbb{G}^m$ is $[\boldsymbol{0}]$, test if $\boldsymbol{x}^\top [\boldsymbol{c}] \overset{?}{=} [0]$ for random $\boldsymbol{x} \in \mathbb{F}_p^m$. The soundness error is $1/p$.

Definition 4.2.5. A **testing distribution** χ_m for $\mathbb{F}_p^m$ is a distribution over $\mathbb{F}_p^m$.

From a testing distribution χ_m, we want that given m independent challenges $\boldsymbol{x}_1, \dots, \boldsymbol{x}_m \leftarrow \chi_m$, the probability that $\boldsymbol{X} = (\boldsymbol{x}_1, \dots, \boldsymbol{x}_m) \in \mathbb{F}_p^{m \times m}$ is invertible is high. Note that $\det(\boldsymbol{X}) \neq 0$, is equivalent to all $\boldsymbol{x}_i$ being linearly independent, and equivalent to $\bigcap_{i=1}^{m} \ker(\boldsymbol{x}_i^\top) = \{0\}$. These interpretations allow to generalise the idea for dual testing distributions later on.

Remark 4.2.6. While we would like to ascribe some measure of "soundness" to a testing distribution, it turns out that for (rewinding-based) extraction, the workings of the extractor strongly determine the soundness error. For example, an extractor may sample challenges with or without repetition. From a purely information-theoretic argument, one might define the soundness error $\delta_{\mathsf{snd}}(\chi)$ of a testing distribution χ as $\sup_{\boldsymbol{0} \neq \boldsymbol{z} \in \mathbb{F}_p^m} \Pr_{\boldsymbol{x} \xleftarrow{\$} \chi_m} [\boldsymbol{x}^\top \boldsymbol{z} = 0]$, i.e. the maximal probably of the test failing to detect a non-zero $\boldsymbol{z}$.

As the information-theoretic soundness measure noted above is a viable sanity check for good testing distributions, we recall the well-known Schwartz–Zippel lemma, which can be used for simple upper bounds on the "information-theoretic soundness error". For this, we use a minor generalisation of the lemma of Schwartz–Zippel.

Lemma 4.2.7 (Schwartz–Zippel). *Let $f \in \mathbb{F}_p[X_1, \dots, X_n]$ be a* non-zero *polynomial of (total) degree d. Let $\mathcal{D}$ be a distribution on $\mathbb{F}_p^n$. Let $p_\infty(\mathcal{D}) := \sup_{\boldsymbol{y} \in \mathbb{F}_p} \Pr[\boldsymbol{x} = \boldsymbol{y} \mid x \xleftarrow{\$} \mathcal{D}]$, Then $\Pr_{\boldsymbol{x} \xleftarrow{\$} \mathcal{D}}[f(\boldsymbol{x}) = 0] \leq dp^{n-1} \cdot p_\infty(\mathcal{D})$*

Proof. Suppose $f \neq 0$ has degree d. By the usual Schwartz–Zippel lemma, we find $\Pr_{\boldsymbol{x} \xleftarrow{\$} \mathbb{F}_p^n}[f(\boldsymbol{x}) = 0] \leq d/p$. In particular, the vanishing set $V(f) := \{\boldsymbol{x} \mid f(\boldsymbol{x}) = 0\} \subseteq \mathbb{F}_p^n$ has at most dp^{n-1} elements. Thus, $\Pr_{\boldsymbol{x} \xleftarrow{\$} \mathcal{D}}[f(\boldsymbol{x}) = 0] = \Pr_{\boldsymbol{x} \xleftarrow{\$} \mathcal{D}}[\boldsymbol{x} \in V(f)] \leq \#V(f) p_\infty(\mathcal{D}) \leq dp^{n-1} p_\infty(\mathcal{D})$, as claimed. □

Example 4.2.8 (Polynomial/Monomial testing). We write χ_m^{mon} for the testing distribution induced by $\boldsymbol{x} = (\xi^0, \dots, \xi^{m-1})$, where $\xi \leftarrow \mathcal{S}$, where $\mathcal{S} \subseteq \mathbb{F}_p^\times$. The distribution induced by the monomials ξ^i, and thus $\boldsymbol{X}$ is a Vandermonde matrix. Hence $\boldsymbol{X}$ is invertible as long as no ξ was chosen twice. Moreover, the information-theoretical soundness error satisfies $\delta_{\mathsf{snd}}(\chi_m^{\mathrm{mon}}) \leq (m-1)/\#\mathcal{S}$.

Remark 4.2.9. Observe that we restricted to $\mathcal{S} \subseteq \mathbb{F}_p^\times$ in Example 4.2.8, i.e. we exclude 0. The reason is, that we want x^{-i} to be well-defined in security proofs.

Example 4.2.10. For the special case $m = 2$, and testing distribution with $\boldsymbol{x} = (\alpha, 1)$ where $\alpha \xleftarrow{\$} \mathcal{S}$ for some $\mathcal{S} \subseteq \mathbb{F}_p$ we write $\chi^{(\beta)}$ and $\alpha \xleftarrow{\$} \chi^{(\beta)}$. If $\mathcal{S} \subseteq \mathbb{F}_p^\times$, i.e. $\alpha \neq 0$, we write $\chi^{(\beta \neq 0)}$. Up to permutation, this distribution equivalent to χ_2^{mon}

Example 4.2.11 (Random testing). The uniform distribution over $\mathbb{F}_p^m$ is a testing distribution. The Lemma of Schwartz–Zippel immediately yields $\delta_{\mathsf{snd}}(\chi) \leq \frac{m}{p}$. Moreover, one can resort to a set $\mathcal{S}$ of "small exponents", i.e. draw from $\mathcal{S} = \{0, \dots, \ell - 1\}$ and still have information-theoretic soundness error at most $\frac{1}{\ell}$.

Example 4.2.12 (Tensor testing). An intermediate choice between monomial and (fully) random testing distributions is offered by tensor-based testing. We denote a tensor-based testing distribution over $\mathbb{F}_p^m = \mathbb{F}_p^{k^\ell} \cong (\mathbb{F}_p^k)^{\otimes \ell}$ by $\chi_k^{\otimes \ell}$, where χ_k is the base testing distribution, and $\boldsymbol{x} \xleftarrow{\$} \chi_k^{\otimes \ell}$ is sampled via $\boldsymbol{x} := \boldsymbol{\xi}_1 \otimes \dots \otimes \boldsymbol{\xi}_\ell$ for $\boldsymbol{\xi}_1, \dots, \boldsymbol{\xi}_\ell \xleftarrow{\$} \chi_k$. One can view tensor-based tests as repeated applications of testing, e.g. $\boldsymbol{\xi}_1 \otimes \boldsymbol{\xi}_2 \cdot \boldsymbol{z} = \boldsymbol{\xi}_1 \cdot (\mathrm{id}_k \otimes \boldsymbol{\xi}_2) \cdot \boldsymbol{z}$. This idea is explained and used in protocols in Section 4.3.3.2, where we can show tighter security guarantees when using tensor-based instead of monomial testing distributions. The information-theoretic soundness error satisfies $\ell \cdot \delta_{\mathsf{snd}}(\chi_k) \leq \delta_{\mathsf{snd}}(\chi_k^{\otimes \ell})$.

Example 4.2.13 (Pseudo-random testing). The verifier can replace truly random choices, e.g. $\boldsymbol{x} \xleftarrow{\$} \mathbb{F}_p^m$, by pseudorandom choices, e.g. $\boldsymbol{x} \xleftarrow{\$} \mathsf{PRG}(s)$ for $s \xleftarrow{\$} \{0,1\}^\lambda$. This allows the verifier to compress such challenges to a random seed s.

It is heuristically plausible, that when using a non-pathological PRG to seed randomness, the resulting testing distribution has an information-theoretic soundness error (negligibly) close to using true randomness. In fact, for a PRG which is secure against *non-uniform* adversaries, this is easy to see.

Note that soundness of testing distributions is a combinatorial property. No pseudorandomness property is required, as illustrated by all other examples. Thus, instead of compressing the randomness to a small seed via a PRG, better (provably secure) constructions with small seeds should exist.

4.2.2.1. Dual Testing Distributions

Testing distributions are essentially a stronger (and simplified) form of the general concept of probabilistic verification with efficient extraction by solving linear systems. They allow to test if an element in $\mathbb{F}_p^n$ is 0. By dualising, we find another concept, for which an intuitive description seems harder. Instead of a distribution on $\boldsymbol{x}^\top \in \mathbb{F}_p^{1\times m}$ where $\sup_{0\neq \boldsymbol{z}\in\mathbb{F}_p^m} \Pr_{\boldsymbol{x}\xleftarrow{\$}\chi_m}[\boldsymbol{z} \in \ker(\boldsymbol{x}^\top)]$ should be small, we consider a distribution on $\boldsymbol{M} \in \mathbb{F}_p^{m\times m-1}$, where $\sup_{0\neq \boldsymbol{z}\in\mathbb{F}_p^m} \Pr_{\boldsymbol{M}\xleftarrow{\$}\chi_m}[\boldsymbol{z} \in \operatorname{im}(\boldsymbol{M})]$ should be small. In a sense, dual testing allows to *enforce* $\boldsymbol{z} = \boldsymbol{0}$ by working in $\operatorname{im}(\boldsymbol{M})$, instead of *testing* $\boldsymbol{x}^\top\boldsymbol{z} = \boldsymbol{0}$.

More concretely, we can use this to ensure that for a Pedersen commitment $[c] = [\boldsymbol{G}|\boldsymbol{H}](\begin{smallmatrix}\boldsymbol{w}\\ \boldsymbol{z}\end{smallmatrix})$ the adversary must have $\boldsymbol{z} = \boldsymbol{0}$. We do so by constructing $[\boldsymbol{H}]$ as $[\boldsymbol{H}] \coloneqq [\boldsymbol{H}']\boldsymbol{M}$. Intuitively, knowledge of some $[c'] = [\boldsymbol{G}|\boldsymbol{H}'](\begin{smallmatrix}\boldsymbol{w}\\ \boldsymbol{y}\end{smallmatrix})$ cannot be transferred to $[\boldsymbol{G}|\boldsymbol{H}]$ because we must have $\boldsymbol{z} = \boldsymbol{M}\boldsymbol{y}$, i.e. $\boldsymbol{z} \in \operatorname{im}(\boldsymbol{M})$, which is unlikely (except for $\boldsymbol{z} = \boldsymbol{0}$ or if $\mathcal{A}$ breaks the binding property). Thus, we can provably "zero" a part of a commitment without an (expensive) argument, simply by changing the commitment key. Generally, this allows to derive "fresh" commitment keys. Using this is more communication efficient than picking and sending a fresh $[\boldsymbol{H}] \leftarrow \mathbb{G}^m$.

Morally, dual testing *enforces* $\boldsymbol{z} = \boldsymbol{0}$, while "normal" testing *verifies* $\boldsymbol{z} = \boldsymbol{0}$.

Definition 4.2.14. An (arbitrary) **dual testing distribution** $\chi_m^\vee$ is a distribution on $\mathbb{F}_p^{m\times(m-1)}$. The information-theoretic soundness error is defined as $\sup_{0\neq \boldsymbol{z}\in\mathbb{F}_p^m} \Pr_{\boldsymbol{M}\xleftarrow{\$}\chi_m}[\boldsymbol{z} \in \operatorname{im}(\boldsymbol{M})]$

Let χ_m be a testing distribution on $\mathbb{F}_p^m$ such that $\boldsymbol{x} \xleftarrow{\$} \chi_m$ *always* has $x_1 = 1$. Then $\chi_m^\vee$ defined as follows is a dual testing distribution: To pick $\boldsymbol{M} \xleftarrow{\$} \chi_m^\vee$, pick $\boldsymbol{x}^\top = (1, \boldsymbol{x}')^\top \xleftarrow{\$} \chi_m$ and let $\boldsymbol{M} \coloneqq \boldsymbol{M}_{\boldsymbol{x}} \coloneqq \begin{pmatrix}\boldsymbol{x}'\\ -\mathrm{id}_{m-1}\end{pmatrix}$. By construction $\ker(\boldsymbol{x}^\top) = \operatorname{im}(\boldsymbol{M}_{\boldsymbol{x}})$, and consequently $\delta_{\mathsf{snd}}(\chi_m^\vee) = \delta_{\mathsf{snd}}(\chi_m)$.

Note that by construction, $\boldsymbol{M}_{\boldsymbol{x}}$ is the (parity) check matrix for the linear code with generator $\boldsymbol{x}$. In particular, $\boldsymbol{x}^\top\boldsymbol{M}_{\boldsymbol{x}} = \boldsymbol{0}$. For simplicity, we only consider dual testing distributions associated to some testing distribution. Also note that we use the information-theoretic soundness error only as a sanity check. Security proofs of protocols rely on special soundness.

4.2.3. Special Soundness, Revisited

Recall that, to prove knowledge soundness, we use the notion of special soundness. In this chapter, we rely very explicitly on linearity and effectively all challenges are viewed as vectors in $\mathbb{F}_p^n$. A challenge vector $\boldsymbol{x}$ may be given directly as an element in $\mathbb{F}_p^n$. But the vector may also be derived from an underlying challenge, e.g. for monomial testing distributions the challenge $\xi \leftarrow \mathbb{F}_p$ is expanded into $(1, \xi, \dots, \xi^{n-1}) \in \mathbb{F}_p^n$. In any case, extraction requires the (derived) challenge vectors to be linearly independent. Observe that, *distinct* challenge *vectors* are not necessarily linearly independent; however, distinct *derived* challenge vectors may imply linear independence, e.g. for monomial testing the ξ_i are distinct iff the derived $\boldsymbol{x}_i$ are linearly independent. To capture this behaviour, we change the definition of valid trees as follows: A tree of challenges is **valid** if for every node, any subset of (at most) n sibling challenge vectors are linearly independent.[7] We note that, as with distinct challenges, basic tree-finders for such trees are easy to construct.

Moreover, following interpretation of trees of challenges will turn out to be useful.

[7] Observe that for, e.g. for monomial testing, this implies distinct choices $\xi_1, \dots, \xi_k$, even if $k > n$ challenges are considered. More generally, one can define special soundness w.r.t. a monotone access structure on the challenges.

Remark 4.2.15. In some situations, we can think of a verifier sending multiple challenges in a single round (e.g. this is the case for tensor-based testing distributions, Example 4.2.12). To indicate that this constitutes a special tree structure, we parenthesize such "multiple" challenges, e.g. both $(2, 3, 3)$ and $\mu = (2, (3, 3))$ denote a tree of size 18, but in the former the verifier sends 3 challenges, while in the latter, it sends 2 challenges (with the second challenge having a "special" structure). We provide a more detailed and concrete explanation in Section 4.3.3.2 where we first encounter this in our protocols.

4.2.3.1. Short-Circuit Extraction

In this section, we assume TreeFind produces the tree's nodes and leaves on demand, and Ext queries TreeFind *as an oracle*, and *traverses the tree in depth-first order*. Moreover, we are in a situation where Ext either extracts a witness for some statement, or a solution to a (supposedly) hard problem, or both. Concretely, we have statements like "we extract $\boldsymbol{w}$ such that either $[\boldsymbol{g}]\boldsymbol{w} = [c]$ is a valid commitment opening, or $[\boldsymbol{g}]\boldsymbol{w} = [0]$ breaks the hard kernel assumption for $[\boldsymbol{g}]$."

Definition 4.2.16. Consider the setting of $\boldsymbol{\mu}$-special soundness (Section 2.5). Suppose $\mathcal{R}$ is an OR-relation $\mathsf{OR}(\mathcal{R}_1, \mathcal{R}_2)$, i.e. $\mathcal{R} = \{((\mathbb{x}_1, \mathbb{x}_2), (i, \mathbb{w}_i)) \mid (\mathbb{x}_i, \mathbb{w}_i) \in \mathcal{R}_i, i \in \{1, 2\}\}$.

Suppose there is some $\boldsymbol{\mu}' \leq \boldsymbol{\mu}$, that is $\mu'_i \leq \mu_i$ for all i, such that extractor Ext has following property. For any valid $\boldsymbol{\mu}$-tree *tree*, $\mathsf{Ext}(\mathbb{x}, \textit{tree})$ we have either:

- Ext explores the tree in a depth-first manner.
- Quick-extraction: Ext finishes exploring a node in layer ℓ, after μ'_ℓ children (i.e. subtrees) under this node are explored (and Ext has recovered a partial witness for this node),
- Short-circuit extraction: Whenever Ext explores more than μ'_i children (without success),[8] it finishes after exploring all μ_ℓ children of that node (or earlier) and returns a witness for $\mathbb{x}_2$.
- If short-circuit does not occur, then Ext returns a witness for $\mathbb{x}_1$ (after exploring a μ'-subtree of *tree*) If short-circuit extraction occurs, a larger subtree may be explored.

We say that such an Ext has **short-circuit** extraction for finding a witness to $\mathbb{x}_1$ or to $\mathbb{x}_2$, or more precisely, has μ'-quick μ-short extractability (for $\mu' \leq \mu$).

Caution 4.2.17. Short-circuit extraction is *not symmetric*. The order of the relation $\mathcal{R}_1$, $\mathcal{R}_2$ in the OR-statement matters! We always think of the second one as the "hard relation" whose witness will lead to a short-circuit.

Remark 4.2.18. While it is often the case that a μ-special sound protocol is also a μ'-quick μ''-short extractable protocol, it can happen that $\mu'' \neq \mu$ (and the relaxed soundness relations may differ)! An example of this appears in Section 4.3.3.1. We do not know whether this is a mere proof artefact or (non-pathological) separating examples exist, hence we strictly separate special soundness and short-circuit extraction in our claims.

Our definition is ad-hoc and tailored to our needs. We leave a general definition and precise treatment of short-circuit extraction for future work.

[8] If $\mu'_i = \mu_i$, then no short-circuitting is possible in this layer.

Corollary 4.2.19. *If* Ext *as in Definition 4.2.16 traverses a valid tree tree in depth-first order, we have following "runtime" guarantees: Let $\boldsymbol{\mu}' = (\mu'_1, \dots, \mu'_n) \leq (\mu_1, \dots, \mu_n) = \boldsymbol{\mu}$. In case of quick-extraction, at most $\prod_{i=1}^{n} \mu'_i$ leaves are explored. In case of short-circuit extraction, at most $s_1 + 1$ leaves are explored, where $s_1 = \sum_{i=1}^{n} (\mu_i - 1) \prod_{j=i+1}^{n} \mu'_j$. In particular, $s_1 \leq (\sum_{i=1}^{n} \mu_i)(\prod_{i=1}^{n} \mu'_i)$.*

Observe that if $\boldsymbol{\mu}' = (\mu', \dots, \mu')$ resp. $\boldsymbol{\mu} = (\mu, \dots, \mu)$ are fixed to μ' resp. μ in each layer, then if $\mu > \mu'$ and $\mu' \neq 1$, we get

$$s_1 = (\mu - 1) \sum_{j=1}^{n} (\mu')^{j-1} = (\mu - 1) \frac{(\mu')^n - 1}{\mu' - 1}.$$

In more generality, let $M'_i = \prod_{j=i}^{n} \mu'_i$, and $a = \#\{i \mid \mu'_i = 1\}$. We find

$$s_1 = \sum_{i=1}^{n} (\mu_i - 1) M'_{i+1} = \sum_{i=1}^{n} \frac{\mu_i - 1}{\mu'_i} M'_i \leq \max_i \Big(\frac{\mu_i - 1}{\mu'_i} \Big) \sum_{i=1}^{n} M'_i \leq \max_i \Big(\frac{\mu_i - 1}{\mu'_i} \Big) (1 + a) M'_1$$

Hence, it is useful to keep all μ'_i and the ratios $\frac{\mu_i}{\mu'_i}$ small, in order to minimize $1 + s_1$, the worst-case number of visited leaves for short-circuit extraction.

Proof of Corollary 4.2.19. Let s_i denote the *maximal number of leaves* necessary to ensure a $\mu|_i$-subtree, where $\mu|_i = (\mu_i, \dots, \mu_n)$, is extractable. We define $s_{n+1} = 1$ and find $s_n = \mu_n = (\mu_n - 1) \cdot 1 + 1$. Recursively, we find $s_i = (\mu_i - 1) \prod_{j=i+1}^{n} \mu'_j + s_{i+1}$. For this, we argue as follows.

In the worst case, layer i short-circuits. If that happens, we have to extract all μ_i nodes. A subtree (in layer $i + 1$) quick-extracts after exploring $\prod_{j=i+1}^{n} \mu'_j$ leaves. In case of failure of quick-extraction, the subtree must short-circuit, requiring at most s_{i+1} nodes. In the worst case, the first $\mu_i - 1$ nodes in layer i quick-extract, and the last node, i.e. the μ_i-th node, again short-circuits. Thus, we again pay the costs[9] for a short-circuit extraction, now in layer $i + 1$, which is bounded by s_{i+1}. Hence, at most s_{i+1} nodes are explored.

To derive the formula for s_1, let $M'_i = \prod_{j=i}^{n} \mu'_j$ and observe that by induction,

$$s_1 = (\mu_1 - 1) M'_2 + s_2 = \dots = \sum_{i=1}^{n} (\mu_i - 1) M'_{i+1} + s_n$$

where $s_{n+1} = 1$ by definition (and $M_{n+1} = 1$). The claim follows. □

We note that since the tree *tree* is randomised (or Ext might explore children in random order), the above worst-case analysis is rather conservative. In Appendix B.9 we discuss two candidate short-circuit extractors. Unfortunately, the candidates currently only have conjectured and heuristic runtime bounds, respectively, but they lack formal proofs.

[9] Since $\mu' \leq \mu$, short-circuit extraction is never cheaper.

4.3. HVZK Arguments for $[A]w = [t]$

We set up the notation for the rest of this section. Let $ck := [\boldsymbol{g}] = [g_0, \overline{\boldsymbol{g}}] \xleftarrow{\$} \mathbb{G}^{1\times n+1}$ be a Pedersen commitment key, where $[g_0] \in \mathbb{G}$ and $[\overline{\boldsymbol{g}}] \in \mathbb{G}^n$. Define $\mathrm{Com}_{\boldsymbol{g}}(\boldsymbol{w}; r) := [g_0]r + [\overline{\boldsymbol{g}}]\boldsymbol{w}$ for $r \in \mathbb{F}_p, \boldsymbol{w} \in \mathbb{F}_p^n$. In some subsections, we let $[\boldsymbol{g}] \in \mathbb{G}^{1\times n}$, i.e. there is no special randomness term g_0; the dimensions will always be clear from the context. We work with matrices $[\boldsymbol{A}] \in \mathbb{G}^{m\times n}$, and vectors $\boldsymbol{w} \in \mathbb{F}_p^n$ and $[\boldsymbol{t}] \in \mathbb{G}^m$, with the these dimensions unless otherwise specified. The target (witness) relation $\mathcal{R}$ is $\mathcal{R} = \{(([\boldsymbol{A}], [\boldsymbol{t}]), \boldsymbol{w}) \mid [\boldsymbol{A}]\boldsymbol{w} = [\boldsymbol{t}]\}$ that is, $\mathbb{x} = ([\boldsymbol{A}], [\boldsymbol{t}])$ and $\mathbb{w} = \boldsymbol{w}$. However, in many cases, this relation is only the correctness guarantee, but the statistical knowledge soundness guarantee (typically in form of special soundness) is for a relaxed relation. For example, some protocols prove that knowledge for either $\exists \boldsymbol{w}\colon [\boldsymbol{A}]\boldsymbol{w} = [\boldsymbol{t}]$ or a non-trivial kernel element for $[\boldsymbol{g}]$.

4.3.1. Intuition

In this section, we devise communication efficient public-coin HVZK arguments for knowledge of a preimage of a linear map, i.e. $\exists \boldsymbol{w}\colon [\boldsymbol{A}]\boldsymbol{w} = [\boldsymbol{t}]$. We follow two principles: "Use probabilistic (batch) verification to check many things at once" and "If messages are too long, replace them by a shorter proof (of knowledge)." For this, we use shrinking commitments, to keep the messages small.

Our strategy is as follows: First, we recall the well-known general HVZK protocol [CD98; Mau15] for proving $\exists \boldsymbol{w}\colon [\boldsymbol{A}]\boldsymbol{w} = [\boldsymbol{t}]$ where $[\boldsymbol{A}] \in \mathbb{G}^{m\times n}$. Then, we show how to apply batch verification to reduce the argument for $([\boldsymbol{A}], [\boldsymbol{t}])$ to another an argument for some $([\boldsymbol{B}], [\boldsymbol{u}])$ with $[\boldsymbol{B}] \in \mathbb{G}^{2\times n}$. This makes communication independent of the number m of rows of $[\boldsymbol{A}]$. We also show how to reduce to a single row, i.e. $m = 1$, but the soundness guarantee must be significantly relaxed.

After this, we revisit the arguments from [BCC+16] which recursively batch statement and witness, i.e. they reduce the number n of columns of $[\boldsymbol{A}]$. Unlike [BCC+16; BBB+18], we need a zero-knowledge version of these arguments. If $[\boldsymbol{A}] = [\boldsymbol{g}]$ is a commitment matrix, we provide a HVZK transformation with constant communication and logarithmic computational overhead. For general (adversarial) $[\boldsymbol{A}]$, we also provide a transformation, which for certain (large) $[\boldsymbol{A}]$ can outperform naive full-blown masking of $\boldsymbol{w}$ at the price of computational soundness and higher complexity; we believe it to be mostly of theoretical interest.

4.3.2. Step 0: A standard Σ-Protocol for $[A]w = [t]$

Here, we recall the prototypical Σ-protocol in a group setting [CD98; Mau15].

Protocol 4.3.1 (Σ_{std}). The following is a protocol to prove $\exists \boldsymbol{w}\colon [\boldsymbol{t}] = [\boldsymbol{A}]\boldsymbol{w}$, using testing distribution $\chi^{(\beta)}$ for challenges, cf. Example 4.2.10. Common input is $([\boldsymbol{A}], [\boldsymbol{t}]) \in \mathbb{G}^{m\times n} \times \mathbb{G}^n$. The prover's witness is some $\boldsymbol{w} \in \mathbb{F}_p^n$.

- P $\to$ V: Pick $\boldsymbol{r} \xleftarrow{\$} \mathbb{F}_p^n$ and compute $[\boldsymbol{a}] = [\boldsymbol{A}]\boldsymbol{r}$. Send $[\boldsymbol{a}] \in \mathbb{G}^m$.
- V $\to$ P: Pick and send $\beta \xleftarrow{\$} \chi^{(\beta)}$.
- P $\to$ V: Compute $\boldsymbol{z} = \beta\boldsymbol{w} + \boldsymbol{r}$. Sends $\boldsymbol{z} \in \mathbb{F}_p^n$.
- V: Check $[\boldsymbol{A}]\boldsymbol{z} \overset{?}{=} \beta[\boldsymbol{t}] + [\boldsymbol{a}]$. Accept/reject if true/false.

It is straightforward to show that any $(x_1, x_2) \xleftarrow{\$} \chi_2$ can be used instead of $\chi^{(\beta)}$, as long as $x_2 \neq 0$, so that $x_1 w + x_2 r$ is uniformly distributed, cf. Section 4.1.1.

Lemma 4.3.2. *Protocol Σ_{std} is a HVZK-PoK for $\exists w\colon [t] = [A]w$. It is perfectly complete, has perfect HVZK and is 2-special sound.*

Proof. **Completeness:** is straightforward to verify.

Extraction: We are given two accepting transcripts $([a], \beta, z)$, and $([a], \beta', z')$ with $\beta - \beta' \neq 0$. Due to the final check of the verifier, we obtain $\frac{1}{\beta-\beta'}[A](z - z') = [t]$. Consequently, $w := \frac{1}{\beta-\beta'}(z - z')$ is a witness.

HVZK: Pick $\beta \leftarrow \chi^{(\beta)}$ and $z \leftarrow \mathbb{F}_p^m$. Then $[a] := [A]z - \beta[t]$ is uniquely defined. Since the distribution of β and z is as in an honest execution, this yields a perfect simulation. □

Now, we improve communication efficiency. We do this in two steps. First, we make the communication independent of the number m of equations, using batch-verification. Then we make it logarithmic in the size n of the witness, using techniques from [BCC+16; BBB+18]. We apply all techniques mentioned in the introduction, using shrinking commitments to keep messages small. Composition of proof systems is implicit due the following remark.

Remark 4.3.3. AND-proofs for statements of the form $\exists w\colon [A]w = [t]$ are trivial. Namely, to prove $\exists w\colon [A_1]w = [t_1] \wedge [A_2]w = [t_2]$, it suffices to define $[A] = \left[\begin{smallmatrix} A_1 \\ A_2 \end{smallmatrix}\right]$ and $[t] = \left[\begin{smallmatrix} t_1 \\ t_2 \end{smallmatrix}\right]$ and prove $\exists w\colon [A]w = [t]$. This AND-compilation technique will be used without explicit mention.

4.3.3. Step 1: Batching All Equations Together

In this step, we devise a HVZK-AoK for $\exists w\colon [A]w = [t]$, where P's communication is independent of m, the "number of equations". Thus, we have to shrink the message $[a] \in \mathbb{G}^m$ somehow. There is a well-known [FS87] optimization in Σ-protocols which achieves this, assuming that the verifier can re-compute the first prover message from challenge and response. Namely, the first message $[a]$ is completely replaced by the hash $\mathsf{Hash}([a])$ of a collision-resistant hash function (CRHF) Hash. This technique is less useful in our setting, as an important step to achieve succinctness is to avoid sending the (large) response by proving knowledge of a response which the verifier would accept. This does not work well when the verifier's checks are "non-algebraic", such as a CRHF. Thus, our approach is slightly different.

To reduce the size of the first message, we would like to batch all m linear equations (given by $[A]$) into a single linear equation, i.e. replace $[A]$ by a random linear combination of its rows. While we can prove a sometime useful sufficient form of relaxed soundness, we do not know whether this is standard sound or not. Nevertheless, if P has explicitly committed to the witness w (or $[a]$), the statement — excluding the commitment — can be batched, as P cannot change its mind anymore.

Note that the value $[t]$ does not, in general, *bind* the adversary to some fixed w, since the adversary may supply (parts of) $[A]$ in the soundness experiment. Thus, he may know dlogs and generate preimages of $[t]$ freely. By adding a commitment to w, we get around this problem.

By using a shrinking commitment to w, we ensure that the communication is small. Now the verifier can send batching randomness, and a HVZK-AoK for the batched statement is carried out. We directly apply AND-compilation in the protocol. We use general testing distributions, but the reader may want to imagine the familiar setting of polynomial testing with $x = (x^0, \dots, x^m)$ first.

Protocol 4.3.4 (Protocol $\mathsf{LMPA}_{\mathsf{batch}}$). The following is a protocol to prove $\exists \boldsymbol{w}\colon [\boldsymbol{t}] = [\boldsymbol{A}]\boldsymbol{w}$. Let χ_m and $\chi^{(\beta)}$ be testing distributions. Common input is $([\boldsymbol{A}], [\boldsymbol{t}]) \in \mathbb{G}^{m\times n} \times \mathbb{G}^m$. The prover's witness is some $\boldsymbol{w} \in \mathbb{F}_p^n$.

- $\mathsf{P} \longrightarrow \mathsf{V}$: Pick $r_{\boldsymbol{w}} \xleftarrow{\$} \mathbb{F}_p$, and compute $[c_{\boldsymbol{w}}] = [g_0]r_{\boldsymbol{w}} + [\overline{\boldsymbol{g}}]^\top \boldsymbol{w} = \mathsf{Com}(\boldsymbol{w}; r_{\boldsymbol{w}})$. Send $[c_{\boldsymbol{w}}]$.
- $\mathsf{V} \longrightarrow \mathsf{P}$: Pick and send $\boldsymbol{x} \xleftarrow{\$} \chi_m$.
  Let $[\widehat{\boldsymbol{A}}] = \boldsymbol{x}^\top[\boldsymbol{A}] \in \mathbb{G}^{1\times n}$ and $[\widehat{t}] = \boldsymbol{x}^\top[\boldsymbol{t}] \in \mathbb{G}$ be the batched statement (for both P and V). Let $[\boldsymbol{B}] := \begin{bmatrix} g_0 & \overline{\boldsymbol{g}} \\ 0 & \widehat{\boldsymbol{A}} \end{bmatrix}$ and let $\exists(\boldsymbol{w}, r_{\boldsymbol{w}})\colon [\boldsymbol{B}]\begin{pmatrix} r_{\boldsymbol{w}} \\ \boldsymbol{w} \end{pmatrix} = \begin{bmatrix} c_{\boldsymbol{w}} \\ \widehat{t} \end{bmatrix} =: [\boldsymbol{u}]$ be the new statement.
- $\mathsf{P} \leftrightarrow \mathsf{V}$: Engage in Protocol Σ_{std} for $\exists\left(\begin{smallmatrix} r_{\boldsymbol{w}} \\ \boldsymbol{w} \end{smallmatrix}\right)\colon [\boldsymbol{B}]\left(\begin{smallmatrix} r_{\boldsymbol{w}} \\ \boldsymbol{w} \end{smallmatrix}\right) = [\boldsymbol{u}]$.

In words, Protocol $\mathsf{LMPA}_{\mathsf{batch}}$ batches $[\boldsymbol{A}]$ to $[\widehat{\boldsymbol{A}}]$, and carries out an AND-proof for opening the commitment $[c_{\boldsymbol{w}}]$ and that the content $\boldsymbol{w}$ of $[c_{\boldsymbol{w}}]$ is preimage of $[\widehat{t}]$ under $[\widehat{\boldsymbol{A}}]$. This is proven via a subprotocol call to Protocol Σ_{std}.

Lemma 4.3.5. *Protocol* $\mathsf{LMPA}_{\mathsf{batch}}$ *is a* 5*-move HVZK-AoK for* $\exists \boldsymbol{w}\colon [\boldsymbol{t}] = [\boldsymbol{A}]\boldsymbol{w}$ *with* $(m, 2)$*-special soundness for finding a witness or a non-trivial kernel element for* $[\boldsymbol{g}]$*. It is* $(1, 2)$*-quick* $(m, 2)$*-short extractable.*

Proof. **Completeness:** It is straightforward to see that completeness holds.

Zero-knowledge: The simulator picks $\beta, \boldsymbol{x}$ according to the distributions. The simulator proceeds in two steps. First, simulate the Protocol Σ_{std}, i.e. the final three rounds. Since those are now simulated independently of $[c_{\boldsymbol{w}}]$, it picks $[c_{\boldsymbol{w}}] \xleftarrow{\$} \mathbb{G}$ uniformly. This gives a perfect HVZK simulation.

Extraction: Given a valid $(m, 2)$-tree *tree*, we first extract the second layer (i.e. the subprotocol Σ_{std}). If not all of the (sub)extractions yield the same $(r_{\boldsymbol{w}}, \boldsymbol{w})$, we found a non-trivial kernel element for $[\boldsymbol{g}]$ and are finished. So suppose that for all $\boldsymbol{x}_i$, we have $[\boldsymbol{B}_i]\left(\begin{smallmatrix} r_{\boldsymbol{w}} \\ \boldsymbol{w} \end{smallmatrix}\right) = \left[\begin{smallmatrix} c_{\boldsymbol{w}} \\ \widehat{t}_i \end{smallmatrix}\right]$, where the subscript i denotes the matrices of the i-th round. Then in particular,

$$\boldsymbol{x}_i^\top[\boldsymbol{A}]\boldsymbol{w} = [\widehat{\boldsymbol{A}}_i]\boldsymbol{w} = [\widehat{t}_i] = \boldsymbol{x}_i^\top[\boldsymbol{t}]. \tag{4.3.1}$$

Let $\boldsymbol{X} = (\boldsymbol{x}_1, \dots, \boldsymbol{x}_n)$. since *tree* is *valid*, $\boldsymbol{X}$ is invertible. Arranging the m linear equations from Eq. (4.3.1), we find

$$\boldsymbol{X}^\top[\boldsymbol{A}]\boldsymbol{w} = \boldsymbol{X}^\top[\boldsymbol{t}] \quad \text{and hence} \quad [\boldsymbol{A}]\boldsymbol{w} = [\boldsymbol{t}].$$

Thus $\boldsymbol{w}$ is a valid witness. This proves (unconditional) $(m, 2)$-special soundness. To see $(1, 2)$-quick $(m, 2)$-short extractability, consider what happens if the first subextraction $(r_{\boldsymbol{w}}, \boldsymbol{w})$ does not satisfy $[\boldsymbol{A}]\boldsymbol{w} = [\boldsymbol{t}]$. As we just argued that given m identical subextractions, the equation $[\boldsymbol{A}]\boldsymbol{w} = [\boldsymbol{t}]$ must hold. Hence we must find a *distinct* pair $(r'_{\boldsymbol{w}}, \boldsymbol{w}')$ among the m pairs, and thus a non-trivial kernel element to $[\boldsymbol{g}]$. Short-circuit-extraction follows as claimed. □

Remark 4.3.6 (Commitment extending). When working with adversarial $[\boldsymbol{A}]$ (and $[\boldsymbol{t}]$), one can not rely on any hardness assumptions w.r.t. $[\boldsymbol{A}]$. Extending $[\boldsymbol{A}]$ to some $[\boldsymbol{B}]$ which has hardness (as in Protocol 4.3.4) is one way to address this by introducing hardness. For the sake of referencing, we call this *commitment extending* $[\boldsymbol{A}]$.

4.3.3.1. Batching to a Single Row

The batch-verification in Protocol 4.3.4 effectively leads to a protocol for $\exists w\colon [A]w = [t]$ which is computationally sound based on the kernel assumption for $[g]$. More concretely, extraction either yields a witness w with $[t] = [A]w$ or $[g]\binom{r_w}{w} = [0]$. The extraction crucially relied on the fact that the commitment row was excluded from the batching. Now, we consider the case where all rows are batched into a single one. In particular, there is no commitment step and the verifier starts the protocol with the batching randomness.

Remark 4.3.7. For modularity, we present $\mathsf{LMPA}_{\text{bat-sing}}$ without built-in zero-knowledge. For simplicity, we reduce $[A]$ to a single row. If $[A]$ is a matrix of matrices in $(\mathbb{G}^{\mu\times\nu})^{m\times n}$, then reduction to $(\mathbb{G}^{\mu\times\nu})^{1\times n}$ follows completely analogously, see also Remark 4.3.14.

Protocol 4.3.8 (Protocol $\mathsf{LMPA}_{\text{bat-sing}}$). Let χ_m^{mon} be a monomial testing distribution. Consider following protocol. Common input is $([A],[t]) \in \mathbb{G}^{m\times n} \times \mathbb{G}^m$. The prover's witness is some $w \in \mathbb{F}_p^n$ with $[A]w = [t]$.

- $\mathsf{V} \to \mathsf{P}$: Pick and send $x \xleftarrow{\$} \chi_m^{\text{mon}}$.
  Let $[\widehat{A}] = x^\top[A] \in \mathbb{G}^{1\times n}$ and $[\widehat{t}] = x^\top[t] \in \mathbb{G}$ be the batched statement (for both P and V).
- $\mathsf{P} \to \mathsf{V}$: Send w.
- V: Check if $[\widehat{A}]w = [\widehat{t}]$.

We stress that it is crucial that in Protocol $\mathsf{LMPA}_{\text{bat-sing}}$, the verifier only checks $[\widehat{A}]w = [\widehat{t}]$ (even though it could check $[A]w = [t]$), because this is what would be the case if the final check were replaced by, say Σ_{std}, to obtain a zero-knowledge protocol.

Lemma 4.3.9. *For $[A]$ write $[\vec{A}] = \begin{bmatrix} a_1 \\ \vdots \\ a_m \end{bmatrix} \in \mathbb{G}^{m\times n}$. Moreover, let $[a] = [a_1 \dots a_m] = [\vec{A}]^\top$. Protocol $\mathsf{LMPA}_{\text{bat-sing}}$ is a 2-move argument system with correctness relation $\{(([A],[t]),w) \mid [A]w = [t]\}$. It is m-special sound for relaxed soundness relation "$\exists u_i \in \mathbb{F}_p^{m\cdot n}\colon [t_i] = [a]u_i$ for all i". Moreover, for relaxed soundness relation "$\exists w\colon [t] = [A]w$ or a non-trivial kernel element in $[a]$" it is 1-quick and $(m+1)$-short extractable.*

Note that the relations for correctness, relaxed soundness w.r.t. to special soundness, and relaxed soundness w.r.t. short-circuit extraction differ. Indeed, short-circuit extraction requires (up to) $m+1$ transcripts whereas special soundness only requires m. Also observe that we only consider χ_m^{mon}, and not a generic testing distribution. The reason will become clear in the proof. It is related to the reduction to special soundness; basing security on a general knowledge extractor can circumvent this.

Proof. Correctness is trivial. We show special soundness. First, we consider a valid m-tree of purported witnesses v_i with

$$x_i^\top[\vec{A}]v_i = x_i^\top[t] = [t]^\top x_i$$

for $i = 1,\dots,m$. Unlike Lemma 4.3.5, we cannot conclude that either $v_i = v_j$ or a non-trivial kernel relation is found. Thus, we have to argue differently. Namely, let $x \otimes v$ denote the vector of vectors $\begin{pmatrix} x_1 v \\ \vdots \\ x_m v \end{pmatrix} \in (\mathbb{F}_p^n)^m$. Then

$$x_i^\top[\vec{A}]v_i = [\vec{A}]^\top(x_i \otimes v_i) = [\vec{A}]^\top p_i$$

where $\boldsymbol{p}_i := \boldsymbol{x}_i \otimes \boldsymbol{v}_i$ for $i = 1, \dots, m$. Arranging $\boldsymbol{x}_i \in \mathbb{F}_p^m$ into $\boldsymbol{X} = (\boldsymbol{x}_1, \dots, \boldsymbol{x}_m) \in \mathbb{F}_p^{m\times m}$ and arranging $\boldsymbol{p}_i \in (\mathbb{F}_p^n)^m$ into $\boldsymbol{P} = (\boldsymbol{p}_1, \dots, \boldsymbol{p}_m) = \begin{pmatrix} X_{1,1}\boldsymbol{v}_1 & \dots & X_{1,m}\boldsymbol{v}_m \\ \vdots & & \vdots \\ X_{m,1}\boldsymbol{v}_1 & \dots & X_{m,m}\boldsymbol{v}_m \end{pmatrix} \in (\mathbb{F}_p^n)^{m\times m}$, we find

$$[\boldsymbol{A}]^\top \boldsymbol{P} = [\boldsymbol{t}]^\top \boldsymbol{X} \quad \text{and hence} \quad [\boldsymbol{A}]^\top \boldsymbol{P}\boldsymbol{X}^{-1} = [\boldsymbol{t}]$$

Consequently, for $\boldsymbol{W} = \boldsymbol{P}\boldsymbol{X}^{-1}$ and $\boldsymbol{u}_i := \boldsymbol{P}\boldsymbol{X}^{-1}\boldsymbol{e}_i$ we find $[\boldsymbol{t}_i] = [\boldsymbol{a}]\boldsymbol{u}_i$ as claimed.

Note that in $\boldsymbol{W} = \boldsymbol{P}\boldsymbol{X}^{-1}$ the matrices $\boldsymbol{P}$ and $\boldsymbol{W}$ are matrices of vectors, i.e. they lie in $(\mathbb{F}_p^n)^{m\times m}$. Now, we derive the structure of $\boldsymbol{W} = \boldsymbol{P}\boldsymbol{X}^{-1}$ by considering an $(m+1)$-th transcript. By construction, we find for *any* $(m+1)$-th accepting transcript with challenge $\boldsymbol{x}$ then $\boldsymbol{v}$ must either satisfy the equality

$$\boldsymbol{W}\boldsymbol{x} = \boldsymbol{x} \otimes \boldsymbol{v}$$

or a non-trival kernel element of $[\boldsymbol{a}]$ can be computed (via $\boldsymbol{W}\boldsymbol{x} - \boldsymbol{x}\otimes\boldsymbol{v}$). From now on, we assume the equality always holds.

Now, consider any $\boldsymbol{\alpha} \in \mathbb{F}_p^m$. Then for $\boldsymbol{W}\boldsymbol{x} = \boldsymbol{x}\otimes\boldsymbol{v}$, we find

$$(\boldsymbol{\alpha}^\top \cdot \boldsymbol{x}) \otimes \boldsymbol{v} = \sum_{i=1}^m \alpha_i \cdot x_i \boldsymbol{v} = \boldsymbol{\alpha}^\top \cdot \begin{pmatrix} x_1\boldsymbol{v} \\ \vdots \\ x_m\boldsymbol{v} \end{pmatrix} = \boldsymbol{\alpha}^\top \cdot (\boldsymbol{x}\otimes\boldsymbol{v}) = \boldsymbol{\alpha}^\top \boldsymbol{W}\boldsymbol{x} = \sum_{i,j=1}^m \boldsymbol{W}_{i,j}\alpha_i x_j$$

where we used block matrix multiplication conventions for $\boldsymbol{\alpha}^\top \cdot (\boldsymbol{x}\otimes\boldsymbol{v})$ and $\boldsymbol{\alpha}^\top\boldsymbol{W}\boldsymbol{x}$, and interpret $\boldsymbol{W} = (\boldsymbol{W}_{i,j}) \in (\mathbb{F}_p^n)^{m\times m}$ as a matrix of matrices. Picking suitable $\boldsymbol{\alpha}$ depending on $\boldsymbol{x}$, namely $\boldsymbol{\alpha} = (0, \dots, 0, x_{\ell+1}, -x_\ell, 0, \dots, 0)^\top$ where the non-zero components are ℓ and $\ell+1$, we find:

$$0 = (\boldsymbol{\alpha}^\top \boldsymbol{x}) \cdot \boldsymbol{v} = \sum_{i,j=1}^m \boldsymbol{W}_{i,j}\alpha_i x_j$$

At this point, it is helpful to observe that the final sum is the sum of the element-wise product of $\boldsymbol{W}$ and $\boldsymbol{\alpha}\boldsymbol{x}^\top$, where

$$\boldsymbol{\alpha}\boldsymbol{x}^\top = \left[\begin{array}{cccccc} 0 & \dots & 0 & 0 & \dots & 0 \\ \vdots & & \vdots & \vdots & & \vdots \\ \hline x_{\ell+1}x_1 & \dots & x_{\ell+1}x_\ell & x_{\ell+1}x_{\ell+1} & \dots & x_{\ell+1}x_m \\ -x_\ell x_1 & \dots & -x_\ell x_\ell & -x_\ell x_{\ell+1} & \dots & -x_\ell x_m \\ \hline \vdots & & \vdots & \vdots & & \vdots \\ 0 & \dots & 0 & 0 & \dots & 0 \end{array}\right]$$

We see that for $\boldsymbol{\alpha}$ as described, we get

$$\begin{aligned} 0 = \sum_{i,j=1}^m \boldsymbol{W}_{i,j}\alpha_i x_j &= \sum_{j=1}^m \boldsymbol{W}_{\ell,j} x_{\ell+1} x_j - \sum_{j=1}^m \boldsymbol{W}_{\ell,j} x_{\ell+1} x_j \\ &= (\boldsymbol{W}_{\ell,\ell} - \boldsymbol{W}_{\ell+1,\ell+1}) x_{\ell+1} x_\ell + \sum_{j\neq\ell} \boldsymbol{W}_{\ell,j} x_{\ell+1} x_j - \sum_{j\neq\ell+1}^m \boldsymbol{W}_{\ell,j} x_{\ell+1} x_j \end{aligned}$$

Now, we exploit that we considered a *monomial* testing distribution χ^{mon}, hence $x_i x_j = x^{i+j-2}$ (for 1-based indexing). Dividing by $x^{\ell-1}$, the degree of $\boldsymbol{\alpha}^\top\boldsymbol{W}\boldsymbol{x}$ in x is m, and it follows that from $m+1$

transcripts with distinct challenges, we can conclude that $W_{\ell,\ell} = W_{\ell+1,\ell+1}$ and $W_{\ell,j} = 0 = W_{\ell+1,j}$ for all $j \neq \ell, \ell+1$. In particular, the matrix (of vectors) W has the form

$$W = \begin{bmatrix} w & 0 & \dots & 0 \\ 0 & w & & 0 \\ 0 & 0 & \ddots & \vdots \\ 0 & 0 & \dots & w \end{bmatrix}$$

and therefore $Aw = [t]$. This completes the proof. □

We note that our proof is modelled after [BCC+16], though we tried to surface some of the relevant properties more clearly.

4.3.3.2. Trading Challenge Size for Extraction Tightness

In protocols $\mathsf{LMPA}_{\text{batch}}$ and $\mathsf{LMPA}_{\text{bat-sing}}$, a monomial testing distribution χ^{mon} is the straightforward choice. As a consequence, at least $\mu \geq m$ transcripts are required for short-circuit extraction, while $\mu' = 1$ is required for quick-extraction. Thus, the quotient $\frac{\mu-1}{\mu'} \geq m-1$ is large, which is detrimental to runtime-tightness of short-circuit extraction. To improve upon this, we modify the approach slightly.

Firstly, we observe that one can compose $\mathsf{LMPA}_{\text{bat-sing}}$ sequentially if only a part of the matrix $[A]$ is batched: View $[A]$ as a matrix of matrices $[A] = \left[\begin{smallmatrix} A_1 \\ A_2 \end{smallmatrix}\right]$ and similar w and $[t]$ as vectors of vectors. Suppose the dimension of $[t]$ is a power of 2, say $m = 2^\ell$. Then $\mathsf{LMPA}_{\text{bat-sing}}$ can be applied, and the height of $[A]$ and $[t]$ is reduced by half. Instead of sending the witness and checking $[\widehat{A}]w = [\widehat{t}]$, one can again recursively apply $\mathsf{LMPA}_{\text{bat-sing}}$ (i.e. the verifier sends another challenge), until $[t]$ is 1-dimensional and $[A] \in \mathbb{G}^{1\times n}$.

This idea corresponds to a different choice of challenges: Let $m = 2^\ell$. The testing distribution now chooses $\xi = (\xi_1, \dots, \xi_\ell) \xleftarrow{\$} \mathcal{S}^\ell$ where $\mathcal{S} \subseteq \mathbb{F}_p^\times$ and sets $x_i = \xi^{\vec{i}}$, where $\vec{i} \in \{0,1\}$ denotes the binary representation of $i \in \{0, \dots, 2^\ell - 1\}$ and $\xi^{\vec{i}} := \prod_{\ell=1}^{m} \xi_j^{i_\ell}$ (i.e. the usual short-hand notation for multivariate monomials). We write $\chi_2^{\otimes \ell}$ for this testing distribution, since this is in fact the tensor-based testing distribution from Example 4.2.12. Note that, the $x^\top[A]$ and $x^\top[t]$ results in exactly the same final $[\widehat{A}]$ and $[\widehat{t}]$ as we would obtain in the above recursive composition of $\mathsf{LMPA}_{\text{bat-sing}}$. The only difference is, that we did not explicitly think of the verifier sending $\xi_1, \dots, \xi_\ell$ as multiple challenges, sent one after another, but as a single challenge.

Reminder 4.3.10. Recall our notation (from Remark 4.2.15) to indicate such "structured" challenges, which result in a deeper tree shape even though they correspond to a single challenge message in terms of round complexity, we put them in parenthesis, e.g. $(3,2)$- resp. $((3,3),2)$- resp. $((3,3),(2,2))$-short extractability all refer to a protocol with 2 challenges sent in terms of round complexity, but tree height of 2, resp. 3, resp. 4.

An almost immediate consequence of these specially structured challenges is the following.

Corollary 4.3.11. *Consider the situation of Lemma 4.3.9, but with Protocol* $\mathsf{LMPA}_{\text{bat-sing}}$ *using* $\chi_m = \chi_2^{\otimes \ell}$ *as defined above. Then the protocol is* $((1,\dots,1))$*-quick* $((3,\dots,3))$*-short extractable for relaxed soundness relation "*$\exists u_i \in \mathbb{F}_p^{m\cdot n}\colon [t_i] = [a]u_i$ *for all i".*

Proof. The claim follows by taking the recursive composition point of view explained above, and by observing that a non-trivial relation $[\vec{\widehat{A}}]^\top \widehat{\boldsymbol{u}} = [\boldsymbol{0}]$ at any intermediate recursion also yields a relation $[\boldsymbol{a}]\boldsymbol{u} = [0]$ in a straightforward manner. □

Using $\chi_2^{\otimes \ell}$ as in Corollary 4.3.11 instead of monomial testing as in Lemma 4.3.9 increases the challenge size (namely, ℓ elements in $\mathbb{F}_p^\times$ instead of 1) but is otherwise an almost universal improvement: Let $m = 2^\ell$ and $\mathcal{S} \subseteq \mathbb{F}_p^\times$. Then the (proven) knowledge error is $\frac{2\ell}{\#\mathcal{S}}$ instead of $\frac{m-1}{\#\mathcal{S}}$, and the required number of transcripts for short-circuit extraction is at most $2\ell + 1$ instead of m. Thus, already for $m \geq 8$ using $\chi_2^{\otimes \ell}$ (as described) instead of χ_m^{mon} only improves the security.

More generally, by using a different bases, e.g. $m = 16^\ell$ or $m = \sqrt{m}^2$ or even more generally, different sequences $(\mu_1', \dots, \mu_\ell')$, different tradeoffs are possible w.r.t. increase of challenge size and improvement in extraction tightness.

4.3.4. Intermezzo: Batch Proofs of Knowledge

By applying the "linear combination of protocols" technique, to multiple "trivial proofs of knowledge" (cf. Fig. 4.2) we obtain batch verification of statements $([\boldsymbol{A}], [\boldsymbol{t}_i])$, $i = 1, \dots, N$, i.e. multiple statements with fixed matrix $[\boldsymbol{A}]$ but varying $[\boldsymbol{t}]$ as in the setting of [PBD07], in a straightforward way. As a clarifying terminology, we suggest to follow [HHK+17] and use *internal batching* for batching techniques which relate to a single statement, e.g. those presented in Section 4.3.3 above or Section 4.3.5 below. For batching multiple statements into one, as in this section, we suggest *external batching* [HHK+17].

Protocol 4.3.12. The following is a protocol to prove: $\exists \boldsymbol{w}_i \colon [\boldsymbol{A}]\boldsymbol{w}_i = [\boldsymbol{t}_i]$ for $i = 1, \dots, N$. Let χ_{N+1} be a testing distribution, such that $\boldsymbol{x} = (x_0, \dots, x_N) \xleftarrow{\$} \chi_{N+1}$ has $x_0 = 1$ always. Common input is $([\boldsymbol{A}], ([\boldsymbol{t}_i])_i) \in \mathbb{G}^{m \times n} \times \mathbb{G}^n$. The prover's witness are some $\boldsymbol{w}_i \in \mathbb{F}_p^n$ with $[\boldsymbol{A}]\boldsymbol{w}_i = [\boldsymbol{t}_i]$ for all i.

- P ⟶ V: Pick $\boldsymbol{r} \xleftarrow{\$} \mathbb{F}_p^n$ and let $[\boldsymbol{t}_0] = [\boldsymbol{A}]\boldsymbol{r}$. Send $[\boldsymbol{t}_0] \in \mathbb{G}^m$.
- V ⟶ P: Pick and send $\boldsymbol{x} \xleftarrow{\$} \chi_{N+1}$.
- P ⟶ V: Compute $\boldsymbol{z} = \boldsymbol{x}^\top \begin{pmatrix} \boldsymbol{r} \\ \boldsymbol{w}_1 \\ \dots \\ \boldsymbol{w}_N \end{pmatrix} = \boldsymbol{r} + \sum_{i=1}^N x_i \boldsymbol{w}_i$. Send $\boldsymbol{z} \in \mathbb{F}_p^n$.
- V: Check $[\boldsymbol{A}]\boldsymbol{z} \overset{?}{=} [\boldsymbol{t}_0] + \sum_{i=1}^N x_i [\boldsymbol{t}_i]$, and accept/reject if true/false.

Lemma 4.3.13. *Protocol 4.3.12 is a HVZK-PoK for $\exists w_i \colon [\boldsymbol{t}_i] = [\boldsymbol{A}_i]\boldsymbol{w}$ for $i = 1, \dots N$. It is perfectly complete, has perfect HVZK and is $(N+1)$-special sound.*

Proof. **Completeness** is straightforward. **Extraction** uses $N + 1$ accepting transcripts $([\boldsymbol{t}_0], \boldsymbol{x}_j, \boldsymbol{z}_j)$. Let $[\boldsymbol{T}] := [\boldsymbol{t}_0, \dots, \boldsymbol{t}_N]$ and $\boldsymbol{Z}$, $\boldsymbol{X}$ be appropriate matrices built from the $N + 1$ transcripts. Since $[\boldsymbol{A}]\boldsymbol{Z} = [\boldsymbol{T}]\boldsymbol{X}$, we find $(\boldsymbol{r}, \boldsymbol{w}_1, \dots, \boldsymbol{w}_N) := \boldsymbol{Z}\boldsymbol{X}^{-1}$ is a valid witness. For **HVZK** note that $x_0 = 1$ and hence $\boldsymbol{z}$ is uniformly distributed for any honest execution. Thus, we can pick $\boldsymbol{z} \leftarrow \mathbb{F}_p^m$ and let $[\boldsymbol{t}_0] := [\boldsymbol{A}]\boldsymbol{z} - \sum_{i=1}^N [\boldsymbol{t}_i] x_i$ as usual. □

If we squint a bit, we can recast this as a special case of $\mathsf{LMPA}_{\text{bat-sing}}$:

$$\begin{aligned}
[\boldsymbol{A}](\boldsymbol{r} + \sum_{i=1}^{N} x_i \boldsymbol{w}_i) &= [\boldsymbol{Ar}, \boldsymbol{Aw}_1, \dots, \boldsymbol{Aw}_N] \cdot \boldsymbol{x} = \boldsymbol{x}^\top \cdot \begin{bmatrix} \boldsymbol{Ar} \\ \cdots \\ \boldsymbol{Aw}_N \end{bmatrix} = \boldsymbol{x}^\top \cdot \begin{bmatrix} \boldsymbol{A} & & \\ & \ddots & \\ & & \boldsymbol{A} \end{bmatrix} \begin{bmatrix} \boldsymbol{r} \\ \vdots \\ \boldsymbol{w}_N \end{bmatrix} \\
[\boldsymbol{t}_0] + \sum_{i=1}^{N} x_i [\boldsymbol{t}_i] &= [\boldsymbol{t}_0, \dots, \boldsymbol{t}_N] \cdot \boldsymbol{x} = \boldsymbol{x}^\top \begin{bmatrix} \boldsymbol{t}_0 \\ \vdots \\ \boldsymbol{t}_N \end{bmatrix}
\end{aligned}$$

A priori, the relaxed extraction case in Lemma 4.3.9 for $\operatorname{diag}([\boldsymbol{A}], \dots, [\boldsymbol{A}])$ produces "only" witnesses $\vec{\boldsymbol{u}}_i = (\boldsymbol{v}_0^i, \vdots, \boldsymbol{v}_N^i)^\top \in (\mathbb{F}_p^n)^{N+1}$ with $[\boldsymbol{A}, \dots, \boldsymbol{A}]\vec{\boldsymbol{u}}_i = [\boldsymbol{t}_i]$. However, the block diagonal structure $\operatorname{diag}([\boldsymbol{A}], \dots, [\boldsymbol{A}])$ ensures that $\boldsymbol{w}_i = \sum_{i=0}^{N} \boldsymbol{v}_j^i$ satisfies $[\boldsymbol{A}]\boldsymbol{w} = [\boldsymbol{t}_i]$.

The linear combination approach also yields efficient k-out-of-N proofs, by having the verifier only partially fix the challenge. However, this requires care to not become unsound, see [HG13].

While we now have the tools to (in a sense modularly) derive a log-size linear map preimage argument via "batching the witness", we will skip such motivation and directly consider to the optimized approach of [BCC+16; BBB+18]. For completeness, we include the derivation in Appendix B.1.

4.3.5. Step 2: "Batching" the Witness

In this section, we show how to "batch" the witness, i.e. proving $\rtimes \boldsymbol{w}\colon [\boldsymbol{A}]\boldsymbol{w} = [\boldsymbol{t}]$ for $[\boldsymbol{A}] \in \mathbb{G}^{m \times n}$ with communication sublinear in n. For the introduction, one may assume $m = 1$, e.g. $[\boldsymbol{A}] = [\boldsymbol{g}]$. Indeed, this is the most interesting case, as it used for the inner product argument we construct later. Moreover, by an application of $\mathsf{LMPA}_{\text{batch}}$, we can always reduce to $m = 2$ after commitment extending $[\boldsymbol{A}]$.

Remark 4.3.14. It is possible to reduce to $m = 1$ *conceptually*. Namely, let $\mathbb{H} := \mathbb{G}^m$. Then $[\boldsymbol{A}]$ and $[\boldsymbol{t}]$ can be interpreted as $[\boldsymbol{A}] \in \mathbb{H}^{1 \times n}$, $[\boldsymbol{t}] \in \mathbb{H}$, and $[\boldsymbol{A}]\boldsymbol{w} = [\boldsymbol{t}]$. Using $\mathbb{H}$ means working over a (base) vector space of *dimension* $m > 1$. In particular, to draw a random $[b] \xleftarrow{\$} \mathbb{H}$ one now needs a *basis* $[h_i]$ of $\mathbb{H}$ and sets $[b] = \sum r_i [h_i]$ for $r_i \xleftarrow{\$} \mathbb{F}_p$.

4.3.5.1. The General Idea

We present the technique of [BCC+16], but in our situation and notation. For the motivation, we let $m = 1$ and ignore zero-knowledge. For $m > 1$, the argument applies without change (by Remark 4.3.14).

Let $k \in \mathbb{N}$ be the size-reduction of the witness, which we want to achieve per iteration Assume for simplicity that $k|n$, i.e. $n/k \in \mathbb{N}$.[10] We will reduce the equation $[\boldsymbol{A}]\boldsymbol{w} = [\boldsymbol{t}]$ to $[\widehat{\boldsymbol{A}}]\widehat{\boldsymbol{w}} = [\widehat{\boldsymbol{t}}]$, where $[\widehat{\boldsymbol{A}}] \in \mathbb{G}^{1 \times n/k}$, $\widehat{\boldsymbol{w}} \in \mathbb{F}_p^{n/k}$, $[\widehat{\boldsymbol{t}}] \in \mathbb{G}$. To do so, divide $[\boldsymbol{A}]$ and $\boldsymbol{w}$ into k blocks size of size n/k, obtaining vectors/matrices of vectors/matrices i.e. $[\boldsymbol{A}] = [\boldsymbol{A}_1 | \dots | \boldsymbol{A}_k] \in (\mathbb{G}^{1 \times n/k})^{1 \times k}$ with $[\boldsymbol{A}_i] \in \mathbb{G}^{1 \times n/k}$, and likewise $\boldsymbol{w} = \begin{pmatrix} \boldsymbol{w}_1 \\ \cdots \\ \boldsymbol{w}_k \end{pmatrix} \in (\mathbb{F}_p^{n/k})^k$.[11] We want to prove

$$\sum_{i=1}^{k} [\boldsymbol{A}_i]\boldsymbol{w}_i = [\boldsymbol{t}].$$

[10] Pad $[\boldsymbol{A}]$ and the witness with zeroes if necessary. Security proofs now require minor adaptions, see Remark 4.2.2.

[11] It also may be helpful to think of the vector space $(\mathbb{F}_p^{n/k})^k$ as $\mathbb{F}_p^k \otimes \mathbb{F}_p^{n/k}$.

Despite similarities, the techniques from Section 4.3.3 are not applicable, as they only reduce m. The trick of [BCC+16] is to embed our problem into a different one which can be batch-verified. Namely, we exploit that the scalar product is the sum of the diagonal entries (i.e. the trace) of the outer product:

$$\begin{bmatrix} \boldsymbol{A}_1 \\ \vdots \\ \boldsymbol{A}_k \end{bmatrix} (\boldsymbol{w}_1, \dots, \boldsymbol{w}_k) = \begin{bmatrix} \boldsymbol{A}_1\boldsymbol{w}_1 & \boldsymbol{A}_1\boldsymbol{w}_2 & \dots & \boldsymbol{A}_1\boldsymbol{w}_k \\ \boldsymbol{A}_2\boldsymbol{w}_1 & \boldsymbol{A}_2\boldsymbol{w}_2 & \dots & \boldsymbol{A}_2\boldsymbol{w}_k \\ \vdots & \vdots & \ddots & \\ \boldsymbol{A}_k\boldsymbol{w}_1 & \boldsymbol{A}_k\boldsymbol{w}_2 & \dots & \boldsymbol{A}_k\boldsymbol{w}_k \end{bmatrix} \in \mathbb{G}^{k\times k} \tag{4.3.2}$$

Now we can send all terms $[\boldsymbol{A}_i]\boldsymbol{w}_j$ to the verifier. Our probabilistic test has to map both $[\boldsymbol{A}]$ and $\boldsymbol{w}$ to a new (smaller) statement. We can do that by multiplying from the left by $\boldsymbol{x} \in \mathbb{F}_p^k$ and from the right by $\boldsymbol{y} \in \mathbb{F}_p^k$ where $\boldsymbol{x}, \boldsymbol{y} \leftarrow \chi_k$. Consequently, we obtain (from associativity)

$$\boldsymbol{x}^\top \left(\begin{bmatrix} \boldsymbol{A}_1 \\ \vdots \\ \boldsymbol{A}_k \end{bmatrix} (\boldsymbol{w}_1, \dots, \boldsymbol{w}_k) \right) \boldsymbol{y} = \underbrace{\left(\boldsymbol{x}^\top \begin{bmatrix} \boldsymbol{A}_1 \\ \vdots \\ \boldsymbol{A}_k \end{bmatrix} \right)}_{:=\sum_i x_i [\boldsymbol{A}_i] \,=:\, [\widehat{\boldsymbol{A}}]} \underbrace{((\boldsymbol{w}_1, \dots, \boldsymbol{w}_k)\boldsymbol{y})}_{:=\sum_i y_i \boldsymbol{w}_i =: [\widehat{\boldsymbol{w}}]} = \underbrace{\sum_{i,j} x_i y_j [\boldsymbol{A}_i]\boldsymbol{w}_j}_{=:[\widehat{t}]}$$

The prover thus sends the (purported) $[\boldsymbol{A}_i]\boldsymbol{w}_j$, denoted $[\boldsymbol{u}_{i,j}]$, and $\widehat{\boldsymbol{w}}$, the shrunk witness. The verifier checks $\sum_i [\boldsymbol{u}_{i,i}] \overset{?}{=} [\boldsymbol{t}]$ and $[\widehat{\boldsymbol{A}}]\widehat{\boldsymbol{w}} \overset{?}{=} [\widehat{t}] = \sum_{i,j} x_i y_j [\boldsymbol{u}_{i,j}]$.

If the $[\boldsymbol{A}_i]$ satisfy a hard kernel assumption, the prover is committed to $\boldsymbol{w}_1, \dots, \boldsymbol{w}_k$. It is not hard to see that given enough transcripts with suitable structure (depending on the testing distribution), one can extract $\boldsymbol{w}$ (or find non-trivial kernel elements.) We will show this for a more efficient special case. All in all, we reduced the statement $([\boldsymbol{A}], [\boldsymbol{t}])$ to $([\widehat{\boldsymbol{A}}], [\widehat{t}])$ which is smaller by a factor of k. This can be applied recursively.

As noted before, this strategy applies verbatim to $[\boldsymbol{A}]\boldsymbol{w} = [\boldsymbol{t}]$ with $[\boldsymbol{A}] \in \mathbb{G}^{m\times n}$, $\boldsymbol{w} \in \mathbb{F}_p^n$ and $[\boldsymbol{t}] \in \mathbb{G}^m$, and it results in $[\widehat{\boldsymbol{A}}] \in \mathbb{G}^{m\times n/k}$, $\widehat{\boldsymbol{w}} \in \mathbb{F}_p^{n/k}$ and $[\widehat{\boldsymbol{t}}] \in \mathbb{G}^m$.

Remark 4.3.15. In Appendix B.1 we explain how to derive a folding-like protocol in two simpler (modular) steps. It turns out that even the optimized version $\mathsf{LMPA}_{\mathrm{noZK}}$ we discuss next is closely related.

4.3.5.2. Refining the Testing Distribution

It turns out, that by a good choice of testing distribution, we can reduce communication. Namely, we can pick testing distributions with $x_i y_j = z_{j-i}$ for all i, j. Then it is sufficient for the verifier to know the sum of the off-diagonals[12] i.e. $[\boldsymbol{u}_\ell] := \sum_{j-i=\ell} [\boldsymbol{A}_i]\boldsymbol{w}_j$ for $\ell = \pm 1, \dots, \pm(k-1)$ (and $[\boldsymbol{u}_0] = [\boldsymbol{t}]$). This follows from $\sum_{j-i=\ell} x_i y_j [\boldsymbol{A}_i]\boldsymbol{w}_j = z_\ell \sum_{j-i=\ell} [\boldsymbol{A}_i]\boldsymbol{w}_j$. We denote the (purported) $\sum_{j-i=\ell} [\boldsymbol{A}_i]\boldsymbol{w}_j$, sent by the prover, as $[\boldsymbol{u}_\ell]$. Note that $[\boldsymbol{u}_0] = [\boldsymbol{t}]$ need not be sent. From the testing distribution $\widetilde{\chi}_{2k-1}$ we require that $z \overset{\$}{\leftarrow} \widetilde{\chi}_{2k-1}$, belongs to a pair $(\boldsymbol{x}, \boldsymbol{y})$. We always implicitly consider $(\boldsymbol{x}, \boldsymbol{y}, z)$ for $\widetilde{\chi}_{2k-1}$, as these values belong together. We leave the formal definition of such generalized testing distributions to the reader.

[12] Any diagonal which is "parallel" to the diagonal (i.e. $(M_{i,j})_{j-i=\ell}$ for some ℓ) is called off-diagonal.

One testing distribution with this property comes from monomials X^i, e.g. $\boldsymbol{x} = (1, x, \dots, x^{k-1})$ and $\boldsymbol{y} = (1, x^{-1}, \dots, x^{-k+1})$. In this case, $z_\ell = x^{-\ell}$. For efficiency, picking $\boldsymbol{x}$ as before, but $\boldsymbol{y} = (x^{k-1}, \dots, x, 1)$ is also interesting, since this preserves small x^i. In this case, $z_\ell = x^{k-1-\ell}$. (We recall that if x is small, then using Horner's scheme, group-operations can take advantage of this, cf. Remark 4.2.3.)

Definition 4.3.16. We write $\widetilde{\chi}^{\mathrm{mon}}_{2k-1}$ for the (generalized) testing distribution with $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ so that $\boldsymbol{x} = (1, x, \dots, x^{k-1})$, $\boldsymbol{y} = (x^{k-1}, \dots, x, 1)$, and $z_\ell = x^{(k-1)-\ell}$.

Remark 4.3.17. Consider any generalized testing distribution with $x_i y_j = z_{j-i} \neq 0$ (for all i, j), i.e. non-zero z_ℓ. Then, *up to being a scalar multiple*, it has the form of Definition 4.3.16. To prove this, we first note that if any x_i or y_i is 0, then $z_0 = x_i y_i = 0$, contradicting our assumption. Hence all x_i, y_j are non-zero. Now, observe that from $x_i y_j = z_{j-i}$ we can deduce $\frac{x_{i+\ell}}{x_i} = \frac{y_0}{y_\ell}$. So in particular $\frac{x_{i+1}}{x_i} = \frac{y_0}{y_1} =: \rho$ holds for all i. Analogously, we find $\frac{y_{j+1}}{y_j} = \rho^{-1}$. Thus, $x_i = x_0 \cdot \rho^i$ and $y_i = y_0 \cdot \rho^{-i}$, as claimed.

With this optimization, we effectively recover and generalize the implicit building block of [BCC+16; BBB+18] as following protocol.

Protocol 4.3.18 (LMPA$_{\mathrm{noZK}}$). The following is a protocol to prove $\exists \boldsymbol{w}\colon [\boldsymbol{t}] = [\boldsymbol{A}]\boldsymbol{w}$. Let $\widetilde{\chi}^{\mathrm{mon}}_{2k-1}$ be the testing distribution from Definition 4.3.16. Common input is $([\boldsymbol{A}], [\boldsymbol{t}]) \in \mathbb{G}^{m \times n} \times \mathbb{G}^m$. We assume $n = k^d$. The prover's witness is some $\boldsymbol{w} \in \mathbb{F}_p^n$ with $[\boldsymbol{A}]\boldsymbol{w} = [\boldsymbol{t}]$.

Recursive step. Suppose $n = k^d > k$.

- *Notation:* Let $[\boldsymbol{A}] =: [\boldsymbol{A}_1, \dots, \boldsymbol{A}_k] =: [\vec{\boldsymbol{A}}] \in (\mathbb{G}^{m \times n/k})^{1 \times k}$, and $\vec{\boldsymbol{w}} := \begin{pmatrix} \boldsymbol{w}_1 \\ \vdots \\ \boldsymbol{w}_k \end{pmatrix} \in (\mathbb{F}_p^{n/k})^k$. Then we get $[\vec{\boldsymbol{A}}^\top] := \begin{bmatrix} \boldsymbol{A}_1 \\ \vdots \\ \boldsymbol{A}_k \end{bmatrix} \in (\mathbb{G}^{m \times n/k})^k$ and $[\vec{\boldsymbol{A}}]\vec{\boldsymbol{w}} = \sum_{i=1}^k [\boldsymbol{A}_i]\boldsymbol{w}_i = [\boldsymbol{t}]$.
- P $\longrightarrow$ V: Compute $[\boldsymbol{u}_\ell] = \sum_{j-i=\ell} [\boldsymbol{A}_i]\boldsymbol{w}_j$. Send $[\boldsymbol{u}_\ell]$ for $\ell = \pm 1, \dots, \pm(k-1)$. ($[\boldsymbol{u}_0] = [\boldsymbol{t}]$ is known to the verifier.)
- V $\longrightarrow$ P: Pick $\boldsymbol{z} \xleftarrow{\$} \widetilde{\chi}^{\mathrm{mon}}_{2k-1}$ with corresponding $\boldsymbol{x}, \boldsymbol{y}$. Send $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$.
- Both parties compute $[\widehat{\boldsymbol{A}}] = \boldsymbol{x}^\top [\vec{\boldsymbol{A}}^\top] = \sum_i x_i [\boldsymbol{A}_i] \in \mathbb{G}^{m \times n/k}$ and $[\widehat{\boldsymbol{t}}] = \boldsymbol{z}^\top [\vec{\boldsymbol{u}}] = \sum_{\ell=-k+1}^{k-1} z_\ell [\boldsymbol{u}_\ell] \in \mathbb{G}$ as the resulting batched statement. Moreover, P computes $\widehat{\boldsymbol{w}} = \vec{\boldsymbol{w}}^\top \boldsymbol{y} = \sum_i \boldsymbol{w}_i y_i$. The protocol may then be recursively restarted, setting $n \leftarrow n/k, \boldsymbol{w} \leftarrow \widehat{\boldsymbol{w}}$, $[\boldsymbol{t}] \leftarrow [\widehat{\boldsymbol{t}}]$, $[\boldsymbol{A}] \leftarrow [\widehat{\boldsymbol{A}}]$.

Base case. Suppose $n \leq k$.

- P $\longrightarrow$ V: Send $\boldsymbol{w}$.
- V: Test if $[\boldsymbol{A}]\boldsymbol{w} \overset{?}{=} [\boldsymbol{t}]$.

See Appendix B.7 for a sketch of the protocol.

For $k = 2$, the base case could also be at $n = 4$, which reduces (only) round complexity (assuming elements in $\mathbb{G}$ and $\mathbb{F}_p$ have the same bitlength). However, this would require special treatment of $k = 2$ in all further protocols and claims, which is why choose not to do this. Indeed, we could describe Protocol LMPA$_{\mathrm{noZK}}$ for general $n = k_1 \dots k_\ell$, as does [BCC+16], but choose not to.

Lemma 4.3.19 (Recursive stepwise extraction)**.** *Consider the situation above. Let $\widetilde{\chi}^{\mathrm{mon}}_{2k-1}$ be the testing distribution from Definition 4.3.16. Let $[\boldsymbol{u}_\ell]$, $[\boldsymbol{A}_i]$, $[\boldsymbol{t}]$, $\boldsymbol{w}_j$ and $[\widehat{\boldsymbol{A}}]$, $[\widehat{\boldsymbol{t}}]$ be defined as above. Then:*

1. Kernel propagation: *Given a non-trivial kernel element of* $[\widehat{A}]$*, we (efficiently) find a non-trivial kernel element of* $[A]$.
2. Special soundness: *Given* $2k-1$ *accepting transcripts with distinct challenges, i.e. an invertible matrix* Z *of challenges, one can extract (unconditionally) a witness* $[A]w = [t]$.
3. Extra consistency: *Given* $2k$ *accepting transcripts with distinct challenges, if the extracted witness from above does not fit w.r.t. the* $[u_\ell]$*, i.e. if an honest prover would send different* $[u_\ell]$ *for* w*, then we find (additionally) a non-trivial kernel element* v*, i.e.* $[A]v = 0$.
4. Short-circuit extraction: *Given* k *accepting transcripts with distinct challenges, a candidate witness* w' *can be computed. If* w' *is not a valid witness or does not satisfy the additional consistency requirements from item 3, i.e. if* $\sum_{j-i=\ell}[A_i]w'_j \neq [u_\ell]$ *for some* ℓ*, then we are guaranteed to find a non-trivial kernel relation from* $2k$ *distinct challenges. In particular, each round has* k*-quick* $2k$*-short extractability.*

Items 1 and 4 ensure $(k,\dots,k)$*-quick* $(2k,\dots,2k)$*-short extractability, whereas item 2 implies* $(2k-1,\dots,2k-1)$*-special soundness.*

Note that, maybe surprisingly, extraction of a witness w with $[A]w = [t]$ is unconditional, i.e. we have a *proof* of knowledge (though consistency of $[u_\ell]$ is not shown unconditionally). The proof is a minor generalisation of [BCC+16; BBB+18].

Proof. Given a non-trivial kernel element $\widehat{w}$ for $x^\top[\vec{A}^\top]$, i.e. $0 = x^\top[\vec{A}^\top]\widehat{w} = \sum_i [A_i]x_i\widehat{w}$, we see that $\widehat{w}_x$ as defined below satisfies $[A]\widehat{w}_x = 0$. Thus, we can recursively "extend" kernel elements to earlier rounds. Now to the interesting case.

Given $2k-1$ transcripts with distinct challenges $z^{(i)}$, we find

$$\begin{aligned}[u_{-k+1},\dots,u_{k-1}]z^{(i)} &= \Big(\sum_{j=1}^{k} x_j^{(i)}[A_j]\Big)\widehat{w}^{(i)} \\ &= \sum_j [A_j]x^{(i)}\widehat{w}^{(i)} \qquad\qquad \text{where} \quad \widehat{w}_x^{(i)} := \begin{pmatrix} x_1^{(i)}\widehat{w} \\ \vdots \\ x_n^{(i)}\widehat{w} \end{pmatrix} \in (\mathbb{F}_p^{n/k})^k. \\ &= [A_1,\dots,A_n]\widehat{w}_x^{(i)}\end{aligned}$$

Note that $\widehat{w}_x^{(i)}$ is a column vector of vectors, and is multiplied with a row vector of matrices. We will sometimes explicate this by writing $[\vec{A}]$ and $\vec{w}$, see the notation Protocol 4.3.18. Thus, we get $[\vec{A}]\vec{w} = [t]$ for the witness $\vec{w}$. For simplicity, the reader may think of the case $m = 1$, $n = k$ where we deal with "normal" vectors and matrices, cf. Remark 4.3.14. To gather all equations in a single linear system, let

$$Z := (z^{(1)},\dots,z^{(2k-1)}) \in \mathbb{F}_p^{(2k-1)\times(2k-1)} \quad \text{and} \quad \widehat{W} := (\widehat{w}_x^{(1)},\dots,\widehat{w}_x^{(2k-1)}) \in (\mathbb{F}_p^{n/k})^{k\times(2k-1)}$$

and note that we obtain

$$[u_{-k+1},\dots,u_{k-1}]Z = [A_1,\dots,A_k]\widehat{W}$$

as the linear system. Note that Z is invertible, since distinct challenges z are automatically also linearly independent. Multiplication by Z^{-1} yields $W := \widehat{W}Z^{-1}$ which satisfies

$$[u_{-k+1},\dots,u_{k+1}] = [A_1,\dots,A_k]W.$$

In particular, numbering columns from $-k+1$ to $k-1$, shows that the ℓ-th column of W is a preimage of u_ℓ. (However, this preimage is under $[A_1,\dots,A_k]$, and hence not necessarily one an honest prover could have produced.) We only care about the preimage of $[u_0] = [t]$, hence the corresponding column yields a witness $\widetilde{w}$ satisfying $[A]\widetilde{w} = [\widehat{t}]$. This shows item 2, i.e. unconditional extraction.

For item 3, we consider the structure of W. Observe that a W obtained from by extracting an honest prover has the structure

$$W = \begin{bmatrix} w_k & w_{k-1} & \dots & w_1 & 0 & \dots & 0 \\ 0 & w_k & \ddots & w_2 & w_1 & \ddots & 0 \\ 0 & 0 & \ddots & \vdots & \ddots & \ddots & 0 \\ 0 & 0 & \ddots & w_k & w_{k-1} & \dots & w_1 \end{bmatrix}$$

This is, because an honest $[u_\ell]$ is the sum of the ℓ-th off-diagonal of $[A_1,\dots,A_k]^\top(w_1,\dots,w_k)$, cf. Eq. (4.3.2). By arguing similar as in the proof of Lemma 4.3.9, i.e. by deriving polynomial equations in ρ of degree $2k$, where $x = (1,\rho,\dots,\rho^{k-1})$, $y = (\rho^{k-1},\dots,1)$, and $z = (1,\rho,\rho^2,\dots,\rho^{k-1})$, we find that either W satisfies the above structure, or the transcripts yield a non-trivial element v such that $[A]v = 0$. This argument is also completely analogous to [BCC+16; BBB+18].

Finally, let us remark the following: Given k distinct challenges, we can compute a candidate $\vec{w}$ via $(w_1,\dots,w_k)Y = (\widehat{w}_1,\dots,\widehat{w}_k)$. If this is a suitable witness, we have quick-extraction. If $[A]w \neq [t]$, we *must* obtain a non-trivial kernel element of $[A]$ from $2k$ transcripts; this yields the claimed short-circuit extraction. More concretely, we argue as follows: Let $\vec{v}^\top$ be $(\widehat{w}_1,\dots,\widehat{w}_k)Y^{-1}$ as described. Suppose that by using all $2k$ transcripts we obtain a witness $\vec{w} \neq \vec{v}$ as in item 3 (if it fails, we're done). By construction of $\vec{w}$ resp. $\vec{v}$, they satisfy $x_i^\top[\vec{A}]^\top(y_i^\top\vec{w}) = x_i^\top[u]$ resp. $x_i^\top[\vec{A}]^\top(y_i^\top\vec{v}) = x_i^\top[u]$ for $i = 1,\dots,k$. Thus, we find $x_i^\top[\vec{A}]^\top(y_i^\top(\vec{w}-\vec{v})) = [0]$. Lastly, for some $i \in \{1,\dots,k\}$ we have $y_i^\top(\vec{w}-\vec{v}) \neq 0$, else $Y(\vec{w}-v) = 0$ and thus, $\vec{w} = v$, a contradiction. Hence find a non-trivial kernel element of $[\vec{A}]$ as claimed. □

Remark 4.3.20. By using different testing distributions and adapting $\mathsf{LMPA}_{\mathrm{noZK}}$ suitably, many protocol variants can be derived. Unfortunately, we did not find a variant which improved upon the choice of $\widetilde{\chi}^{\mathrm{mon}}_{2k-1}$, hence we presented only that. One interesting approach was to increase the symmetries in the (generalized) testing distribution $\widetilde{\chi}_{2k-1}$ further, by considering $z = x \otimes y + y \otimes x$ for monomial x, y as in $\widetilde{\chi}^{\mathrm{mon}}_{2k-1}$. With this, two off-diagonals are summed together as $[u_{\pm\ell}]$, and thus only $k-1$ elements have to be sent. However, without sending additional elements (namely, $k-1$ more), we were unable to turn this idea into a working protocol. Thus, ultimately this did not lead to any improvements.

4.3.5.3. Adding Zero-Knowledge

There are many variations which yield honest-verifier zero-knowledge. The most straightforward one is to run Protocol 4.3.1 (Σ_{std}) and replace sending z by proving $\exists z\colon [A]z = \beta[t] + [a]$ via $\mathsf{LMPA}_{\mathrm{noZK}}$. This is analogous to [BCC+16; BBB+18], where $\mathsf{LMPA}_{\mathrm{noZK}}$ was only (implicitly) used to save communication, and it results in a communication-efficient *proof* of knowledge, which we denote by $\mathsf{LMPA}_{\mathrm{simpleZK}}$; the combination $\mathsf{LMPA}_{\mathrm{batch+simpleZK}}$ denotes a first application $\mathsf{LMPA}_{\mathrm{batch}}$ and then $\mathsf{LMPA}_{\mathrm{simpleZK}}$. One downside of $\mathsf{LMPA}_{\mathrm{simpleZK}}$ is, that computing $[A]r$ for random r is expensive. If the computational overhead for computing $[A]r$ is acceptable — and it often is — using $\mathsf{LMPA}_{\mathrm{simpleZK}}$ a simple and secure choice. If computational resources are limited, it is natural to try to avoid computing $[A]r$. We will show some means for this.

Intuitively, it suffices to blind the prover's responses (instead of the entire witness), and therefore, a logarithmic amount of randomness should suffice. There are two caveats:

1. If $[\boldsymbol{A}] \in \mathbb{G}^{m \times n}$ is a tall matrix, say $m \gg n$, then this intuition (and our constructions) lead to $m\log(n)$ random terms, and $m\log(n) \gg n$. This problem can be mitigated by using commitment-extension (using a commitment key $[\boldsymbol{g}]$) and batching down to $m = 2$ as done in Protocol $\mathsf{LMPA}_{\text{batch}}$. However, computing $[\boldsymbol{g}]\binom{r_w}{\boldsymbol{w}}$ is requires $\mathcal{O}(n)$ exponentiations.
2. Since $[\boldsymbol{A}]$ may be adversarially chosen, using $[\boldsymbol{A}]$ for blinding turns out to lead to technical difficulties, which would make a generic protocol rather complex (if possible at all). Thus, our approach relies on a trusted commitment key to avoid this.

In the rest of this section, we consider the most relevant special case, which will be used and extended in our inner product argument construction in Section 4.4.4.

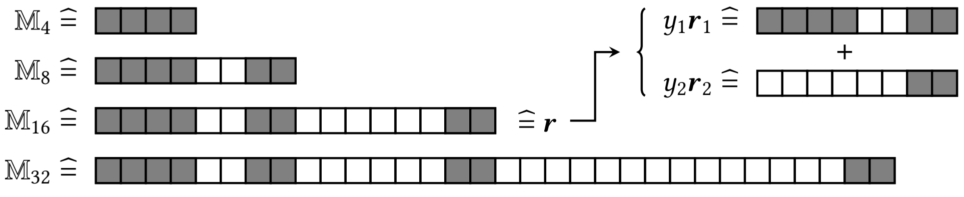
Proving Knowledge of Opening of a Commitment. Suppose that $[\boldsymbol{A}] = [\boldsymbol{g}] \in \mathbb{G}^{1\times n}$, and $[\boldsymbol{g}]$ is a commitment key and $k = 2$. Thus, $[\boldsymbol{A}]$ has hard kernel assumption by construction.

So our current problem is to prove in *zero-knowledge* that $\exists \boldsymbol{w}\colon [\boldsymbol{g}]\boldsymbol{w} = [t]$. We will employ a masked version of $\mathsf{LMPA}_{\text{noZK}}$, with judiciously chosen randomness $\boldsymbol{r}$, to achieve this. In particular, we do not pick $\boldsymbol{r} \xleftarrow{\$} \mathbb{F}_p^n$. We pick $\boldsymbol{r}$ so that only logarithmically many r_i are non-zero. Thus, computing $[\boldsymbol{g}]\boldsymbol{r} = [a]$ is quite cheap (unlike in Protocol Σ_{std}). By the uniform-or-unique guideline, we want that each message $[\boldsymbol{u}_{\pm 1}]$ looks uniformly random. By analysing the recursive structure of $\mathsf{LMPA}_{\text{noZK}}$, we can achieve this by picking $r_i \xleftarrow{\$} \mathbb{F}_p$ for $i \in \mathbb{M}_n \subseteq \{1,\dots,n\}$ with $\mathbb{M}_n$ as defined below, and $r_i = 0$ else.

Definition 4.3.21 (Masking sets). For some implicitly fixed k, we define the **masking (randomness) sets/spaces** $\mathbb{M}_n \subseteq \{1,\dots,n\}$ (for $n = k^d$) by the formulas below. The set $\mathbb{M}_n$ describes the unit vectors of $\mathbb{F}_p^n$ which are used for random masking. We typically treat $\mathbb{M}_n$ as a subvector space of $\mathbb{F}_p^n$ (instead of explicitly referring to its span $\langle \boldsymbol{e}_i \mid i \in \mathbb{M}_n\rangle$).

- $\mathbb{M}_1 \coloneqq \{1\}$ and $\mathbb{M}_k \coloneqq \{1,\dots,k\}$
- $\mathbb{M}_{k^d} \coloneqq \{\mathbb{M}_{k^{d-1}}\} \dot\cup \{ik^{d-1}-1, ik^{d-1} \mid i = 1,\dots,k\}$ for $d \geq 2$.

See Fig. 4.2 for a pictorial description for $k = 2$.



$\mathbb{M}_4 \,\widehat{=}$
$\mathbb{M}_8 \,\widehat{=}$
$\mathbb{M}_{16} \,\widehat{=}$
$\widehat{=}\, \boldsymbol{r}$
$\mathbb{M}_{32} \,\widehat{=}$
$y_1\boldsymbol{r}_1 \,\widehat{=}$
+
$y_2\boldsymbol{r}_2 \,\widehat{=}$

Figure 4.2.: *Left:* The (construction of the) masking randomness sets $\mathbb{M}_4$, $\mathbb{M}_8$, $\mathbb{M}_{16}$ and $\mathbb{M}_{32}$ (for $k = 2$). The squares denote the numbers $1,\dots,n$ (or the respective basis vectors). *Right:* A demonstration of the "overlap" when a recursive step is applied to $\mathbb{M}_{16}$, i.e. $\widehat{\boldsymbol{r}} = y_1\boldsymbol{r}_1 + y_2\boldsymbol{r}_2$ is computed. The indices in $\mathbb{M}_n$ can also be constructed recursively via string concatenation: $m_{2n} = m_n|0^{n-2}11$ and $m_2 = 11$, $m_1 = 1$.

By the structure of the masking sets, we have that (for $k = 2$), if $\boldsymbol{r}$ is split into $\boldsymbol{r} = \binom{\boldsymbol{r}_1}{\boldsymbol{r}_2}$ as in $\mathsf{LMPA}_{\text{noZK}}$, then $[u_{j-i}] = [\boldsymbol{g}_i]\boldsymbol{r}_j$ is uniformly distributed for $\boldsymbol{r} \xleftarrow{\$} \mathbb{M}_n$. Moreover, $\widehat{\boldsymbol{r}} = y_1\boldsymbol{r}_1 + y_2\boldsymbol{r}_2$ is distributed like a fresh $\boldsymbol{r}' \xleftarrow{\$} \mathbb{M}_{n/k}$. Essentially, this holds even when considering the joint distribution $(\widehat{\boldsymbol{r}}, u_{\pm 1},\dots,u_{\pm(k-1)})$. Thus, masking sets exhibit a useful recursive structure.

Protocol 4.3.22. Let $crs = [\boldsymbol{g}] \in \mathbb{G}^{1\times n}$ be a uniformly random commitment key (in particular, $[\boldsymbol{g}]$ has hard kernel relation under the DLOG assumption on $\mathbb{G}$.). The following is a protocol to prove $\exists \boldsymbol{w}\colon [\boldsymbol{g}]\boldsymbol{w} = [t]$. Let $\widetilde{\chi}^{\mathrm{mon}}_{2k-1}$ be the testing distribution from Definition 4.3.16 and $\chi^{(\beta)}$ be from Example 4.2.10. Common input is $(crs, [t]) \in \mathbb{G}^{1\times n} \times \mathbb{G}$. We assume $n = k^d$. The prover's witness is $\boldsymbol{w} \in \mathbb{F}_p^n$ with $[\boldsymbol{g}]\boldsymbol{w} = [\boldsymbol{t}]$.

- P $\rightarrow$ V: Choose $\boldsymbol{r} \xleftarrow{\$} \mathbb{M}_n$. Compute $[a] = [\boldsymbol{g}]\boldsymbol{r}$. Send $[a]$.
- V $\rightarrow$ P: Choose $\beta \xleftarrow{\$} \chi^{(\beta)}$. Send β.
- P $\leftrightarrow$ V: Let $\boldsymbol{z} := \beta\boldsymbol{w} + \boldsymbol{r}$ and $[t'] := \beta[t] + [a]$. Engage in $\mathsf{LMPA}_{\mathrm{noZK}}$ for $\exists \boldsymbol{z}\colon [\boldsymbol{g}]\boldsymbol{z} = [t']$.

Lemma 4.3.23. *Protocol 4.3.22 an AoK for $\exists \boldsymbol{w}\colon [\boldsymbol{g}]\boldsymbol{w} = [t]$ which is perfectly correct, ε-statistical HVZK with $\varepsilon = 2(k-1)\log_k(n)/p$, and $(2, 2k-1, \dots, 2k-1)$-special sound. Moreover is has $(2, k, \dots, k)$-quick $(2, 2k, \dots, 2k)$-short extractability.*

Proof. It is clear that this protocol is correct. Short-circuit extraction follows easily as this is essentially a sequential composition of Protocol Σ_{std} and $\mathsf{LMPA}_{\mathrm{noZK}}$, only the masking randomness differs (which does not affect soundness, since a malicious prover could always pick this randomness). Thus, only zero-knowledge remains.

For HVZK, one should note that $\boldsymbol{z} = \beta\boldsymbol{w} + \boldsymbol{r}$ behaves like a linear combination throughout the protocol, because the reduced witness $\widehat{\boldsymbol{z}}$ is of the form $\beta\widehat{\boldsymbol{w}} + \widehat{\boldsymbol{r}}$. Indeed, we can view the protocol as a linear combination of protocols for $[\boldsymbol{g}]\boldsymbol{w} = [t]$ and $[\boldsymbol{g}]\boldsymbol{r} = [a]$. Thus, we may assume that $\boldsymbol{w} = 0$ and we can focus on $\boldsymbol{r}$ alone. Now, we have to show that in each non-base iteration (i.e. $n > k$), the joint distribution $(\widehat{\boldsymbol{r}}, [u_{-(k-1)}], \dots, [u_{k-1}])$ is (almost) uniform in $\mathbb{M}_{n/k} \times \mathbb{G}^{2(k-1)}$. As explained before, this basically follows due to the form of $\mathbb{M}_n$. A formal proof is given below. For the base case, we note that by construction, $\mathbb{M}_k = \{1, \dots, k\}$. Thus, $\boldsymbol{r} \xleftarrow{\$} \mathbb{M}_k$ is uniformly random in $\mathbb{F}_p^k$, and hence $x\boldsymbol{w} + \boldsymbol{r}$ is uniformly random for $n \le k$, perfectly hiding $\boldsymbol{w}$. In particular, the messages in the base case are uniformly random too. Since the uniform-or-unique property is satisfied, the zero-knowledge simulator can construct the transcript in reverse, as usual.

Now, we prove that in each recursive step, unless a bad event happens, $(\widehat{\boldsymbol{r}}, u_{-(k-1)}, \dots, u_{k-1})$ is distributed uniformly in $\mathbb{M}_{n/k} \times \mathbb{G}^{2(k-1)}$. Recall that we assume w.l.o.g. $\boldsymbol{w} = 0$ and consider the distribution of the execution for $\boldsymbol{r}$ alone. Moreover, for simplicity, we treat the case where $k = 2$. To analyze the distribution of $(\widehat{\boldsymbol{r}}, u_{-1}, u_1)$, we consider the "transition map $\boldsymbol{L}$" (for fixed challenge x) from $\boldsymbol{r} = \binom{\boldsymbol{r}_1}{\boldsymbol{r}_2}$ to $(\widehat{\boldsymbol{r}}, u_{-1}, u_1)$,

$$\underbrace{\begin{pmatrix} x\,\mathrm{id}_2 & \mathrm{id}_2 \\ \hline \boldsymbol{g}_2 & 0 \\ 0 & \boldsymbol{g}_1 \end{pmatrix}}_{=:\boldsymbol{L}} \begin{pmatrix} \boldsymbol{r}_1 \\ \boldsymbol{r}_2 \end{pmatrix} = \begin{pmatrix} \widehat{\boldsymbol{r}} \\ \hline u_{-1} \\ u_1 \end{pmatrix}.$$

Since only the components in $\mathbb{M}_n$ (resp. $\mathbb{M}_{n/k}$) of $\boldsymbol{r}$ (resp. $\widehat{\boldsymbol{r}}$) are non-zero, we can remove all columns not in $\mathbb{M}_n$ and all rows not in $\mathbb{M}_{n/k}$ from $\boldsymbol{L}$, as these consist are entirely of zeroes in $\boldsymbol{r}$ (resp. $\widehat{\boldsymbol{r}}$) and do not contribute anything (for an honest prover). To express the resulting transition, write $\boldsymbol{r}_1'$ for the components in $\mathbb{M}_{n/2} \setminus \{n/2-1, n/2\}$ and $\boldsymbol{r}_1''$ for the components in $\{n/2-1, n/2\}$ of $\boldsymbol{r}_1$, and write $\boldsymbol{r}_2''$ for the components in $\{n/2-1, n/2\}$ of $\boldsymbol{r}_2$ (i.e. the components $\{n-1, n\}$ of $\boldsymbol{r}$). Using a similar convention for $\boldsymbol{g}$ and $\widehat{\boldsymbol{r}}$, and taking into account this recursive structure of $\mathbb{M}_n$, we can now write the transition as

$$\underbrace{\left(\begin{array}{ccc} x\,\mathrm{id}_{\dim(\mathbb{M}_{n/2})-2} & 0 & 0 \\ 0 & x\,\mathrm{id}_2 & \mathrm{id}_2 \\ \hline \boldsymbol{g}_2' & \boldsymbol{g}_2'' & 0 \\ 0 & 0 & \boldsymbol{g}_1'' \end{array}\right)}_{=:\boldsymbol{M}} \begin{pmatrix} \boldsymbol{r}_1' \\ \boldsymbol{r}_1'' \\ \boldsymbol{r}_2'' \end{pmatrix} = \left(\begin{array}{c} \widehat{\boldsymbol{r}}' \\ \widehat{\boldsymbol{r}}'' \\ \hline u_{-1} \\ u_1 \end{array}\right).$$

Since a surjective matrix $\boldsymbol{M}$ maps the uniform distribution to the uniform distribution (Lemma B.4.4), it suffices to prove that $\boldsymbol{M}$ is surjective. By Gaussian elimination, it suffices to prove surjectivity of $\boldsymbol{M}' = \begin{pmatrix} x\,\mathrm{id}_2 & \mathrm{id}_2 \\ \boldsymbol{g}_2'' & 0 \\ 0 & \boldsymbol{g}_1'' \end{pmatrix}$. Equivalently since the matrix is square (for $n \geq 4 = k^2$), it suffices to prove it is invertible or has full rank. It is easy to compute $\det(\boldsymbol{M}')$ and find that it is a (non-zero) polynomial in $\boldsymbol{g}'$ and $\boldsymbol{g}''$ of degree 2 (considering x as a constant), hence by Schwartz–Zippel,[13] except with probability $2/p$ we have $\det(\boldsymbol{M}') \neq 0$. In our case, this is also easily derived by elementary row and column transformations, namely

$$\begin{pmatrix} x\,\mathrm{id}_2 & \mathrm{id}_2 \\ \boldsymbol{g}_2'' & 0 \\ 0 & \boldsymbol{g}_1'' \end{pmatrix} \rightsquigarrow \begin{pmatrix} x\,\mathrm{id}_2 & \mathrm{id}_2 \\ 0 & \frac{1}{x}\boldsymbol{g}_2'' \\ 0 & \boldsymbol{g}_1'' \end{pmatrix} \rightsquigarrow \begin{pmatrix} \mathrm{id}_2 & 0 \\ 0 & \frac{1}{x}\boldsymbol{g}_2'' \\ 0 & \boldsymbol{g}_1'' \end{pmatrix} \rightsquigarrow \begin{pmatrix} \mathrm{id}_2 & 0 \\ 0 & \boldsymbol{g}_2'' \\ 0 & \boldsymbol{g}_1'' \end{pmatrix}$$

and hence it suffices show that $\boldsymbol{g}_1''$ and $\boldsymbol{g}_2''$ are linearly independent, which fails with probability at most $2/p$ since $\boldsymbol{g}_1'', \boldsymbol{g}_2'' \in \mathbb{F}_p^2$ are uniformly random.

We now argue by induction for any $n \geq 2^2$ to extend the claim to multiple rounds. Since $\boldsymbol{g}$ is distributed uniformly by assumption, so are $\boldsymbol{g}_1', \boldsymbol{g}_1'', \boldsymbol{g}_2''$. Hence, $\boldsymbol{g}_1''$ and $\boldsymbol{g}_2''$ are linearly dependent with probability at most $2/p$. Moreover, $\widehat{\boldsymbol{g}} = x\boldsymbol{g}_1 + \boldsymbol{g}_2$ is again uniformly distributed. Thus, by induction, we can upper bound the probability that a non-surjective $\boldsymbol{M}$ is encountered by $2\log_2(n)/p$ over all recursions.

Lastly, for $n = 2$, $\boldsymbol{r}$ completely masks $\boldsymbol{w}$, so the response is uniform as well. Overall, this proves the claim for $k = 2$. For $k > 2$, one argues similarly, but the analysis becomes more technical because $\boldsymbol{M}$ is larger (as it has a row for each $[\boldsymbol{u}_{\pm\ell}]$, $\ell = 1, \ldots, k-1$) and describing it gets more technical. See the proof of Lemma B.4.5 in Appendix B.4 for details. □

Note that, if $[\boldsymbol{g}]$ might be non-uniform or even adversarial, one should use full masking (i.e. Σ_{std}).

Remark 4.3.24. For HVZK, we crucially rely on uniformity of $[\boldsymbol{g}]$. Indeed, if $[\boldsymbol{g}]$ is adversarially chosen, there is an explicit attack. For example, choose $n = 2^3$, and $\boldsymbol{g}$ so that $\boldsymbol{g}_2'' = \alpha\boldsymbol{g}_1''$ (where $\boldsymbol{g}_i''$ are defined as in the proof of Lemma 4.3.23) and let $\widehat{\boldsymbol{w}} = \boldsymbol{0}$. Then it can be shown that $(\widehat{\boldsymbol{r}}, u_{-1}, u_1)$ satisfies a non-trivial relation which is efficiently verifiable given α, namely, $(1+\alpha)[\boldsymbol{g}_1'']\widehat{\boldsymbol{r}}'' = [u_{-1} + \alpha u_1]$. Thus, the terms are not uniform.

Dealing with General $[\boldsymbol{A}]$**.** We briefly sketch how we deal with a general matrix $[\boldsymbol{A}]$. In Remark 4.3.24, we saw an attack on HVZK for adversarial $[\boldsymbol{A}]$ when using masking sets. An even simpler attack is if $[\boldsymbol{A}]$ has only $\boldsymbol{0}$-columns for all indices in $\mathbb{M}_n$, then $[\boldsymbol{A}]\boldsymbol{r} = [\boldsymbol{0}]$, and the masking is obviously useless. To avoid analyzing the structure of $[\boldsymbol{A}]$, trying to make $\mathbb{M}_n$ dynamically depend on $[\boldsymbol{A}]$, we take a slightly different approach.

[13] We exploit HVZK, i.e. that x is stochastically independent of $\boldsymbol{g}'$ and $\boldsymbol{g}''$, to apply Schwartz–Zippel.

The core idea is, for each row i of $[\boldsymbol{A}]$, to run a proof for $[\boldsymbol{h}]\boldsymbol{r}_i$ in parallel and their linearly combined responses are used as response. Here $[\boldsymbol{h}] \in \mathbb{G}^{1\times n}$ is a uniformly random commitment key and we pick $\boldsymbol{r}_i \xleftarrow{\$} \mathbb{M}_n$ as in Protocol 4.3.22.[14] However, this negatively affects soundness, as only a preimage $(\boldsymbol{w}, \boldsymbol{r}_1, \dots, \boldsymbol{r}_m)$ with $[\boldsymbol{A}_i|\boldsymbol{h}]\left(\begin{smallmatrix}\boldsymbol{w}\\ \boldsymbol{r}_i\end{smallmatrix}\right)$ is extracted, in particular $[\boldsymbol{A}]\boldsymbol{w} \neq [\boldsymbol{t}]$ may happen. To counteract this, we use a commitment extension of $[\boldsymbol{A}]$ (with $[\boldsymbol{g}]$), similar to $\mathsf{LMPA}_{\mathrm{batch}}$. Then, either a preimage $\boldsymbol{w}$ with $[\boldsymbol{A}]\boldsymbol{w} = [\boldsymbol{t}]$ or a non-trivial kernel element for $[\boldsymbol{g}|\boldsymbol{h}]$ is found, which yields computational soundness.

The more more complicated nature and the (necessary) fall-back to computational soundness, with the only upside of the HVZK-compilation being (potentially) more computationally efficient than the naive masking of $\mathsf{LMPA}_{\mathrm{simpleZK}}$, lead us to conclude that this result is mostly of theoretical interest. As the outlined approach is rather straightforward, yet its analysis is technical, provides no further insight, and is required for our inner product argument construction in Section 4.4.4, it is relegated to Appendix B.4.

4.3.6. Step 3: Adding (Arithmetic Circuit) Relations to the Witness

If the witness $\boldsymbol{w}$ for $[\boldsymbol{A}]\boldsymbol{w} = [\boldsymbol{t}]$ is committed to, e.g. if the first row of $[\boldsymbol{A}]$ is a Pedersen commitment CRS $[\boldsymbol{g}]$, it is easily possible to make other (zero-knowledge) statements about $\boldsymbol{w}$ by composition of zero-knowledge protocols. Using, for example, Protocol $\mathsf{QESA}_{\mathrm{Copy}}$ from Section 4.4, it is possible to add constraints on the witness. Following slightly more fine-grained approach is often useful.

Remark 4.3.25. Often, $\boldsymbol{w}$ is much larger than the part which has to satisfy some constraints. It is efficiently possible to "split" and "merge" Pedersen commitments i.e. $[c] = [c_1] + [c_2]$ where $[\boldsymbol{G}] = [\boldsymbol{G}_1|\boldsymbol{G}_2]$ and $[c_i] = [\boldsymbol{G}_i]\boldsymbol{w}_i$. With this, one can split off the relevant portion $\boldsymbol{w}_1$ of $\boldsymbol{w}$ into the commitment $[c_1]$ and prove additional relations about this portion only. Splitting is generally very cheap. We use these ideas in our construction of an efficient proof of shuffle of ElGamal cipertexts, see Appendix B.2.1.

4.4. Arithmetic Circuit Satisfiability from Quadratic Equations

In this section, we describe quadratic gates, and relate them to rank 1 constraint systems (R1CS) and arithmetic circuits (AC). Then, we construct a proof of satisfiability of a set of quadratic equations via a (zero-knowledge) inner-product argument.

4.4.1. Quadratic Gates

The equations our scheme is able to prove are *quadratic equations*, i.e. given a witness $\boldsymbol{w} \in \mathbb{F}_p^n$ and a matrix $\boldsymbol{\Gamma} \in \mathbb{F}_p^{n\times n}$ we wish to prove

$$\boldsymbol{w}^\top \boldsymbol{\Gamma} \boldsymbol{w} = 0.$$

We choose this description of quadratic equations for *simplicity* and *uniformity* of notation. In particular, we assume without loss of generality, that the witness $\boldsymbol{w}$ has the constant 1 as first component, i.e. $w_1 = 1$. Our notation is similar to [EG14], which uses such notation for Groth–Sahai proofs [GS08]. Indeed, our arguments are essentially *commit-and-prove* systems [EG14]. Consider a general quadratic equation $\boldsymbol{x}^\top\boldsymbol{\Gamma}\boldsymbol{x} + \boldsymbol{a}^\top\boldsymbol{x} - t = 0$, with $\boldsymbol{a}, \boldsymbol{x} \in \mathbb{F}_p^n$, $\boldsymbol{\Gamma} \in \mathbb{F}_p^{n\times n}$, $t \in \mathbb{F}_p$ with statement given by the constants $(\boldsymbol{a}, \boldsymbol{\Gamma}, t)$. This can be encoded via $\boldsymbol{w} = \left(\begin{smallmatrix}1\\ \boldsymbol{x}\end{smallmatrix}\right)$ and suitably (re)defined $\boldsymbol{\Gamma}$, namely $\boldsymbol{w}^\top\left(\begin{smallmatrix}t & 0\\ \boldsymbol{a} & \boldsymbol{\Gamma}\end{smallmatrix}\right)\boldsymbol{w} = 0$.

[14] For better efficiency, we set all components of $[\boldsymbol{h}]$ outside $\mathbb{M}_n$ to 0 and only pick those in $\mathbb{M}_n$ uniformly.

It is straightforward to encode arithmetic circuits (ACs) as systems of quadratic equations. Doing this allows for ACs built from *quadratic gates*, i.e. gates whose input-output behaviour is described by a quadratic equation.

4.4.2. Arithmetic Circuits and Rank 1 Constraint Systems

Rank 1 constraint systems (R1CS) are systems of equations of the form $(\boldsymbol{w}^\top \boldsymbol{a})(\boldsymbol{b}^\top \boldsymbol{w}) - \boldsymbol{c}^\top \boldsymbol{w} = 0$, where $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c} \in \mathbb{F}_p^n$. Evidently, these are special cases of quadratic equations with $\boldsymbol{\Gamma} = \boldsymbol{a}\boldsymbol{b}^\top + e_1 \boldsymbol{c}^\top$.[15] Arithmetic circuit satisfiability can also be encoded in R1CS. See [BCG+13] for details.

The gates testable by one R1CS equation allow a single "multiplication". As we noted in Section 4.1.2, quadratic equations are more flexible. For example, the inner product $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$ is a single quadratic gate. To the best of our knowledge, n gates are necessary to encode this in R1CS (essentially one per $x_i y_i$ multiplication). Even more extreme, multiplication of secret $n \times n$-matrices requires n^2 quadratic equations and no auxiliary variables. On the other hand, R1CS naively requires n^3 equations and auxiliary variables (though using better matrix multiplication algorithms improves this, e.g. Strassen's algorithm yields $n^{\log_2(8)}$). Thus, in terms of the constraint system alone, QEs can be much more compact than R1CS; but see Appendix B.8 for a discussion of the interplay between constraint system and proof system w.r.t. R1CS and QEs.

Overall, note that quadratic gates enable new optimisations. Indeed, all "AC to R1CS" optimisations (and more), are applicable for "AC to QE".

4.4.3. The Verification Strategy

Verification that a system of quadratic gates is satisfied is easy given the witness $\boldsymbol{w}$, in our case the wire assignments of the AC, and equations $\boldsymbol{\Gamma}_{\mathfrak{g}}$ (the gate $\mathfrak{g}$ encoded as a matrix). One simply checks $\boldsymbol{w}^\top \boldsymbol{\Gamma}_{\mathfrak{g}} \boldsymbol{w} = 0$ for all $\mathfrak{g} \in \mathfrak{G}$. By batching this can be sped up: Pick $(r_{\mathfrak{g}})_{\mathfrak{g}} \overset{\$}{\leftarrow} \chi_{\#\mathfrak{G}}$ from a testing distribution. Then compute $\boldsymbol{\Gamma} := \sum_{\mathfrak{g} \in \mathfrak{G}} r_{\mathfrak{g}} \boldsymbol{\Gamma}_{\mathfrak{g}}$ as the "batched statement". Finally, check if $\boldsymbol{w}^\top \boldsymbol{\Gamma} \boldsymbol{w} = 0$.

We run this strategy in a commit-then-prove manner. First, we commit to the witness $\boldsymbol{w}$. Then we let the verifier pick testing randomness $(r_{\mathfrak{g}})_{\mathfrak{g}}$ and we prove that $\boldsymbol{w}^\top \boldsymbol{\Gamma} \boldsymbol{w} = 0$ where $\boldsymbol{\Gamma} := \sum_{\mathfrak{g} \in \mathfrak{G}} r_{\mathfrak{g}} \boldsymbol{\Gamma}_{\mathfrak{g}}$ is the "batched statement". Note that $\boldsymbol{w}^\top \boldsymbol{\Gamma} \boldsymbol{w} = \langle \boldsymbol{w}, \boldsymbol{\Gamma} \boldsymbol{w} \rangle$ is an inner product. Hence, we require a *zero-knowledge* inner-product argument.

For technical reasons, we cannot generate a commitment to $\boldsymbol{\Gamma w}$ efficiently (prior to knowing $\boldsymbol{\Gamma}$).[16] Therefore, the prover first commits to $\boldsymbol{x} = \boldsymbol{w}$ as $[c_{\boldsymbol{x}}] = \mathsf{Com}_{ck_1}(\boldsymbol{w})$. Then it obtains $\boldsymbol{\Gamma}$ and commits to $\boldsymbol{y} = \boldsymbol{\Gamma w}$ as $[c_{\boldsymbol{y}}] = \mathsf{Com}_{ck_2}(\boldsymbol{\Gamma w})$. Then the prover carries out the inner product argument. It must also *prove* that the commitments $[c_{\boldsymbol{x}}]$ and $[c_{\boldsymbol{y}}]$ open to values $\boldsymbol{x} = \boldsymbol{w}$ and $\boldsymbol{y} = \boldsymbol{\Gamma w}$ as promised. Again, we use (linear) batching to shorten the proof for $\boldsymbol{y} = \boldsymbol{\Gamma x}$. Namely, to check $\boldsymbol{y} = \boldsymbol{\Gamma x}$, the verifier picks random $\boldsymbol{s} \leftarrow \chi_n$ (after receiving $[c_{\boldsymbol{x}}], [c_{\boldsymbol{y}}]$) and the prover proves $0 = \langle \boldsymbol{\Gamma x} - \boldsymbol{y}, \boldsymbol{s} \rangle$. This batch-verification is n-special sound.

[15] In our context, the name "R1CS" may be misleading, since $\boldsymbol{\Gamma}$ can have rank 2, i.e. the rank of $\boldsymbol{\Gamma}$ is ≤ 2 for R1CS (and arbitrary for general quadratic equations). Nevertheless, we follow this standard naming convention.

[16] If $[\boldsymbol{g}]$ is a Pedersen commitment key and $[c] = [\boldsymbol{g}]\boldsymbol{w}$, then $[\boldsymbol{h}] = [\boldsymbol{g}]\boldsymbol{\Gamma}^{-1}$ is a Pedersen commitment key where $[c] = [\boldsymbol{h}](\boldsymbol{\Gamma w})$. We do not use this for various reasons, in particular, since computing $\boldsymbol{\Gamma}^{-1}$ is expensive.

Instead of running two inner product arguments (for $\langle \boldsymbol{\Gamma x} - \boldsymbol{y}, \boldsymbol{s}\rangle = 0$ and $\langle \boldsymbol{x}, \boldsymbol{y}\rangle = 0$) we immediately batch verify again: The verifier picks randomness α and the prover proves knowledge of openings $\boldsymbol{x}, \boldsymbol{y}$ such that,

$$\begin{aligned} \langle \boldsymbol{x} - \alpha \boldsymbol{s}, \boldsymbol{y} + \alpha \boldsymbol{\Gamma}^\top \boldsymbol{s}\rangle &= \langle \boldsymbol{x}, \boldsymbol{y}\rangle + \alpha \left(\langle \boldsymbol{x}, \boldsymbol{\Gamma}^\top \boldsymbol{s}\rangle - \langle \boldsymbol{s}, \boldsymbol{y}\rangle\right) - \alpha^2 \langle \boldsymbol{s}, \boldsymbol{\Gamma}^\top \boldsymbol{s}\rangle \\ &= \langle \boldsymbol{x}, \boldsymbol{y}\rangle + \alpha \langle \boldsymbol{\Gamma x} - \boldsymbol{y}, \boldsymbol{s}\rangle - \alpha^2 \langle \boldsymbol{s}, \boldsymbol{\Gamma}^\top \boldsymbol{s}\rangle \\ &\overset{!}{=} \underbrace{-\alpha^2 \langle \boldsymbol{s}, \boldsymbol{\Gamma}^\top \boldsymbol{s}\rangle}_{=:t} \end{aligned} \tag{4.4.1}$$

where t is fixed by the random choices of the verifier. For fixed $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\Gamma}, \boldsymbol{s}$, this is a degree 2 polynomial, and hence this batch-verification is 3-special sound. the lemma of Schwartz–Zippel can be applied to the polynomial in α. Thus, the overall strategy is (special) sound. To instantiate it, we need a zero-knowledge inner product argument.

4.4.4. Zero-Knowledge Inner Product Argument

Now, we show how to construct a zero-knowledge inner product argument (IPA). We first recall [BCC+16; BBB+18], from a high level. We identify [BBB+18] as a linear combination of protocols. We achieve HVZK similar to Protocol 4.3.22 by masking the witness, but we also exploit the redundancy (or kernel) guideline. Addition of zero-knowledge adds a single round, where one group element and one challenge are sent. (Note that $m = 1$ now.) For technical reasons we have a base case at $n = 8$ (for $k = 2$). In practice, this is hardly worth mentioning.

4.4.4.1. Inner Product Argument (IPA)

First, we describe the (non-zero-knowledge) inner product argument following [BCC+16; BBB+18]. It will also be evident how to extend [BBB+18] to $k > 2$. Since $k = 2$ minimises communication, we only mention this in passing.

Our setting is as follows: We have a CRS $crs = ([\boldsymbol{g}'], [\boldsymbol{g}''], [Q])$ for which finding a non-trivial kernel element of $[\boldsymbol{g}', \boldsymbol{g}'', Q] \in \mathbb{G}^{2n+1}$ is hard. In other words, these are three independent (or one large tripartite) Pedersen commitment keys.

Naively, one proves knowledge of openings of $c'_{\boldsymbol{w}}$ and $c''_{\boldsymbol{w}}$ with $\langle \boldsymbol{w}', \boldsymbol{w}''\rangle = t$. The idea and argument(s) in Section 4.3.5, in particular Protocol 4.3.22, allow to recursively shrink our statement. After one recursion step, we obtain $\langle \widehat{\boldsymbol{w}}', \widehat{\boldsymbol{w}}''\rangle = \widehat{t}$. The prover sends terms $v_{\pm 1} = \langle \boldsymbol{w}'_i, \boldsymbol{w}''_j\rangle$ (for $j - i = \pm 1$), so that the verifier can compute $\widehat{t}$, in analogy to $[u_{\pm 1}]$ in Section 4.3.5,

Let $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) \xleftarrow{\$} \widetilde{\chi}^{\text{mon}}$ be challenges from the generalized testing distribution Definition 4.3.16 (as in Protocols 4.3.18 and B.4.1). A naive linear combination of Protocol $\mathsf{LMPA}_{\text{noZK}}$ for $\boldsymbol{w}'$ and $\boldsymbol{w}''$ does not work well. Namely, we would get $\widehat{t} = \langle \widehat{\boldsymbol{w}}', \widehat{\boldsymbol{w}}''\rangle = \langle \boldsymbol{x}^\top \vec{\boldsymbol{w}}', \boldsymbol{x}^\top \vec{\boldsymbol{w}}''\rangle$, but there are no compatibility guarantees for this expression, and indeed for $\boldsymbol{x} = (1, \xi)$, $\boldsymbol{y} = (\xi, 1)$, we concretely find

$$\langle \widehat{\boldsymbol{w}}', \widehat{\boldsymbol{w}}''\rangle = \langle \boldsymbol{w}'_1, \boldsymbol{w}''_1\rangle + \xi(\langle \boldsymbol{w}'_1, \boldsymbol{w}''_2\rangle + \langle \boldsymbol{w}'_2, \boldsymbol{w}''_1\rangle) + \xi^2 \langle \boldsymbol{w}'_2, \boldsymbol{w}''_2\rangle.$$

In analogy to $[u_0]$ in $\mathsf{LMPA}_{\text{noZK}}$, we want that the term $t = \langle \boldsymbol{w}'_1, \boldsymbol{w}''_1\rangle + \langle \boldsymbol{w}'_2, \boldsymbol{w}''_2\rangle$ appears (perhaps scaled by ξ) and is preserved. Instead, the "mixed terms" are preserved this way. Fortunately, we solved this problem in Section 4.3.5 already. The solution is to use $\langle \boldsymbol{x}^\top \boldsymbol{w}', \boldsymbol{y}^\top \boldsymbol{w}''\rangle$ Thus, we find

$$\langle \boldsymbol{x}^\top \boldsymbol{w}', \boldsymbol{y}^\top \boldsymbol{w}''\rangle = \xi \langle \boldsymbol{w}', \boldsymbol{w}''\rangle + \langle \boldsymbol{w}'_1, \boldsymbol{w}''_2\rangle + \xi^2 \langle \boldsymbol{w}'_2, \boldsymbol{w}''_1\rangle$$

Therefore, we run the protocol for $\boldsymbol{w}'$ with challenge $(\boldsymbol{x},\boldsymbol{y})$, and we run the protocol for $\boldsymbol{w}''$ with flipped challenge $(\boldsymbol{y},\boldsymbol{x})$. Now, as in Protocol $\mathsf{LMPA}_{\mathrm{noZK}}$, it suffices to send $v_{j-i} := \langle \boldsymbol{w}'_i, \boldsymbol{w}''_{2-i}\rangle$ (for $i = 1, 2$).

The argument described above is a hybrid of [BCC+16] and [BBB+18]. For security, we need that "commitment merging" (see Remark 4.3.25), which the linear combination of protocols induces, still is *binding*. To obtain [BBB+18], we simply *commit* to v_ℓ as well (using $[Q]$), and send the combined commitment, i.e. apply again a linear combination. This "merged" commitment key is now $[\boldsymbol{g}', \boldsymbol{g}'', Q]$. Thus instead of sending two messages thrice (namely $[u'_{\pm 1}]$, $[u''_{\mp 1}]$, $[v_{\mp 1}Q]$), we only send the two "merged commitments" $[u_{\pm 1}] = [u'_{\pm 1}] + [u''_{\mp 1}] + [v_{\mp 1}Q]$. Unlike [BBB+18], which uses $\boldsymbol{x} = (\xi^{-1}, \xi)$ we prefer $\boldsymbol{x} = (1, \xi)$ since exponentiation with 1 is free.

Protocol 4.4.1 ($\mathsf{IPA}_{\mathrm{noZK}}$). The following is an inner product argument for relation

$$\mathcal{R}_{\mathsf{IPA}} = \{(([c], t), (\boldsymbol{w}', \boldsymbol{w}'')) \in \mathbb{F}_p^n \mid [c] = [\boldsymbol{g}']\boldsymbol{w}' + [\boldsymbol{g}'']\boldsymbol{w}'' + t[Q] \ \wedge\ \langle \boldsymbol{w}', \boldsymbol{w}''\rangle = t\}.$$

Let $\chi^{(\beta\neq 0)}$ be a testing distribution, and let $\widetilde{\chi}^{\mathrm{mon}}_{2k-1}$ be the generalized testing distribution from Definition 4.3.16, i.e. we have $\boldsymbol{z} \xleftarrow{\$} \widetilde{\chi}^{\mathrm{mon}}_{2k-1}$ (with $\boldsymbol{z}$ indexed from $-k$ to k) together with $\boldsymbol{x}, \boldsymbol{y}$ such that $z_{j-i} = x_i y_j$. Common input is $crs = ([\boldsymbol{g}', \boldsymbol{g}'', Q]) \in \mathbb{G}^{1\times n} \times \mathbb{G}^{1\times n} \times \mathbb{G}$ and the statement $([c], t)$ We assume $n = k^d$. The prover's witness is $(\boldsymbol{w}', \boldsymbol{w}'')$ such that $(([c], t), (\boldsymbol{w}', \boldsymbol{w}'')) \in \mathcal{R}_{\mathsf{IPA}}$.

- $\mathsf{V} \longrightarrow \mathsf{P}$: (Step 0: "Fixing" t.) Pick and send $\alpha \xleftarrow{\$} \chi^{(\beta\neq 0)}$. Both sides set $[Q] := \alpha^{-1}[Q]$. Then they set $[c] := ([c] - \alpha t[Q]) + t[Q]$.[17]

  Recursive step. Suppose $n = k^d > 1$.

- $\mathsf{P} \longrightarrow \mathsf{V}$: Compute $[\boldsymbol{u}'_\ell] = \sum_{i-j=\ell} [\boldsymbol{g}'_i]\boldsymbol{w}'_j$, where $[\boldsymbol{g}'_j]$ and $[\boldsymbol{w}'_j]$ are as usual (i.e. split $[\boldsymbol{g}']$, $[\boldsymbol{w}']$ into k equal-size pieces). Compute the respective $[\boldsymbol{u}''_\ell]$. Let $v_\ell := \sum_{\ell=j-i} \langle \boldsymbol{w}'_i, \boldsymbol{w}''_j\rangle$. Let $[\boldsymbol{u}_\ell] := [\boldsymbol{u}'_\ell] + [\boldsymbol{u}''_{-\ell}] + v_{-\ell}[Q]$. Send $[\boldsymbol{u}_\ell]$ for $\ell = \pm 1, \ldots, \pm(k-1)$.[18]
- $\mathsf{V} \longrightarrow \mathsf{P}$: pick $\boldsymbol{z} \xleftarrow{\$} \widetilde{\chi}_{2k-1}$ with corresponding $\boldsymbol{x}, \boldsymbol{y}$. Send $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$.
- Both parties compute $[\widehat{\boldsymbol{g}}'] = \boldsymbol{x}^\top[\vec{\boldsymbol{g}}'] = \sum x_i[\boldsymbol{g}'_i] \in \mathbb{G}^{1\times n/k}$ and $[\widehat{\boldsymbol{g}}''] = \boldsymbol{y}^\top[\vec{\boldsymbol{g}}''] \in \mathbb{G}^{1\times n/k}$, and $[\widehat{c}] = \boldsymbol{z}^\top[\vec{\boldsymbol{u}}] = \sum_\ell z_\ell[\boldsymbol{u}_\ell] \in \mathbb{G}$ as the new batched statement. Moreover, P computes $\widehat{\boldsymbol{w}}' = \boldsymbol{y}^\top\vec{\boldsymbol{w}}'$ and $\widehat{\boldsymbol{w}}'' = \boldsymbol{x}^\top\vec{\boldsymbol{w}}''$. (Note the invariant: $\widehat{t} = \boldsymbol{z}^\top\boldsymbol{v} = \sum_{\ell=-k+1}^{k-1} z_\ell v_\ell = \langle \widehat{\boldsymbol{w}}', \widehat{\boldsymbol{w}}''\rangle$.)
  Skipping Step 0, recursively continue with $n \leftarrow n/k, \boldsymbol{w}' \leftarrow \widehat{\boldsymbol{w}}', \boldsymbol{w}'' \leftarrow \widehat{\boldsymbol{w}}''$, $[c] \leftarrow [\widehat{c}]$, $[\boldsymbol{g}'] \leftarrow [\widehat{\boldsymbol{g}}']$, $[\boldsymbol{g}''] \leftarrow [\widehat{\boldsymbol{g}}'']$.

  Base case. Suppose $n = 1$.

- $\mathsf{P} \longrightarrow \mathsf{V}$: Send $\boldsymbol{w}', \boldsymbol{w}''$.
- V: Let $t := \langle \boldsymbol{w}', \boldsymbol{w}''\rangle$ and test if $[c] \overset{?}{=} [\boldsymbol{g}']\boldsymbol{w}' + [\boldsymbol{g}'']\boldsymbol{w}'' + t[Q]$.

See Appendix B.7 for a sketch of this protocol.

The above argument is correct by inspection. In Step 0, we "enforce" value t in the $[Q]$-component of the commitment. Hence, t is not explicit in any further computation, but $[c]$ satisfies the invariant that it is a commitment to $\boldsymbol{w}', \boldsymbol{w}'', t$ where $t = \langle \boldsymbol{w}', \boldsymbol{w}''\rangle$.

[17] This changes the committed value s in $[c]$ to $\alpha(s - t) + t$ under the new $[Q]$.
[18] Note that $[\boldsymbol{u}_0]$ is implicitly known to V (as $[c]$).

Lemma 4.4.2. *Protocol* $\mathsf{IPA}_{\mathrm{noZK}}$ *is* $(2, 2k, \ldots, 2k)$*-special (relaxed) sound for finding a witness in* $\mathcal{R}_{\mathsf{IPA}}$ *or a non-trivial element in the kernel of* $[\boldsymbol{g}', \boldsymbol{g}'', Q]$*. Moreover, it is* $(1, k, \ldots, k)$*-quick* $(2, 2k, \ldots, 2k)$*-short extractable for the same relation.*

The proof is straightforward, and essentially the same as [BCC+16; BBB+18].

Proof sketch. First, we ignore Step 0, i.e. we treat t as part of the prover's witness and do not change $[Q]$ or $[c]$. (This is the case in the recursive steps.)

Then, the proof of extraction is essentially the same as Lemma 4.3.19. More concretely: The base case is extractable by definition. In each recursive step, we need at most $2k$ transcripts $\widehat{\boldsymbol{w}}_i', \widehat{\boldsymbol{w}}_i'', \widehat{t}_i, \boldsymbol{z}_i$ (with fixed $[u_\ell]$) which we arrange into a matrix equation analogous to the proof for Lemma 4.3.19. By computing $\boldsymbol{w}', \boldsymbol{w}''$ and t from k transcripts by using $\boldsymbol{Y}^{-1}$ (resp. $\boldsymbol{X}^{-1}$) we get quick-extraction, unless $t \neq \langle \boldsymbol{w}', \boldsymbol{w}'' \rangle$.

If quick-extraction fails, there are two possibilities. Either the sent values $[u_{\pm\ell}]$ (and $[u_0]$) are incompatible with $(\boldsymbol{w}', \boldsymbol{w}'', t)$, or $t \neq \langle \boldsymbol{w}', \boldsymbol{w}'' \rangle$. In the former case, from $2k$ transcripts the binding property is broken, i.e. we must then find a non-trivial element in $\ker([\boldsymbol{g}', \boldsymbol{g}'', \alpha Q_{\mathrm{old}}])$, as in Lemma 4.3.19, and (since $\alpha \neq 0$) we find one in $\ker([\boldsymbol{g}', \boldsymbol{g}'', Q_{\mathrm{old}}])$ (where we write $[Q_{\mathrm{old}}]$ to denote $[Q]$ before it is overwriting in Step 0). Let us now consider the latter case, i.e. $t \neq \langle \boldsymbol{w}', \boldsymbol{w}'' \rangle$ but all messages are consistent with $(\boldsymbol{w}', \boldsymbol{w}'', t)$. Since we have $2k$ transcripts which are consistent with $(\boldsymbol{w}', \boldsymbol{w}'', t)$, we find

$$\langle \widehat{\boldsymbol{w}}', \widehat{\boldsymbol{w}}'' \rangle = \langle \boldsymbol{x}^\top \boldsymbol{w}', \boldsymbol{y}^\top \boldsymbol{w}'' \rangle = \sum_{\ell=k-1}^{k+1} \sum_{j-i=\ell} \langle \boldsymbol{w}_i', \boldsymbol{w}_j'' \rangle z_\ell = \sum_\ell v_\ell z_\ell$$

for all $2k$ challenges. This implies that $t = v_0 = \langle \boldsymbol{w}', \boldsymbol{w}'' \rangle$, so the case $t \neq \langle \boldsymbol{w}', \boldsymbol{w}'' \rangle$ cannot happen (unless the binding property was broken, which did not happen by assumption).

Finally, let us consider Step 0. Given 2 transcripts with challenges $\alpha_1 \neq \alpha_2$ and extracted witnesses (from subtrees) $\boldsymbol{w}'$, $\boldsymbol{w}''$, and $s := \langle \boldsymbol{w}', \boldsymbol{w}'' \rangle$. These witnesses are identical for both subtrees, or we find a non-trival kernel element. But then the recomputation of $[c]$ and $[Q]$ in Step 0 implies that $\alpha_1(s - t) = \alpha_2(s - t)$, which implies $s - t = 0$ as claimed. □

4.4.4.2. Adding Zero-Knowledge

Making the inner-product argument zero-knowledge can be done in many ways. To be competetive with Bulletproofs [BBB+18], which uses the IPA without zero-knowledge, we directly mask the witness (as in Protocol 4.3.22, unlike $\mathsf{LMPA}_{\mathsf{ZK}}$). This is problematic, since the scalar product is non-linear. Consequently, our (initial) approach only works under some (mild) constraints.

As mentioned above, the problem with using masking randomness and proving $\langle \beta\boldsymbol{w}' + \boldsymbol{r}', \beta\boldsymbol{w}'' + \boldsymbol{r}'' \rangle$ is the non-linearity:

$$\langle \beta\boldsymbol{w}' + \boldsymbol{r}', \beta\boldsymbol{w}'' + \boldsymbol{r}'' \rangle = \langle \boldsymbol{r}', \boldsymbol{r}'' \rangle + \beta(\langle \boldsymbol{r}', \boldsymbol{w}'' \rangle + \langle \boldsymbol{w}', \boldsymbol{r}'' \rangle) + \beta^2 \langle \boldsymbol{w}', \boldsymbol{w}'' \rangle$$

Thus, to linearize we need to send $\langle \boldsymbol{r}', \boldsymbol{r}'' \rangle$ and $m = \langle \boldsymbol{r}', \boldsymbol{w}'' \rangle + \langle \boldsymbol{w}', \boldsymbol{r}'' \rangle$, But, we also need to show that $\langle \boldsymbol{r}', \boldsymbol{w}'' \rangle + \langle \boldsymbol{w}', \boldsymbol{r}'' \rangle$ does not reveal anything about $\boldsymbol{w}'$ or $\boldsymbol{w}''$. Instead of doing this directly, we pick randomness with the "kernel guideline" in mind, namely

- $\boldsymbol{r}'$ is chosen freely, and
- $\boldsymbol{r}''$ is chosen subject to $\langle \boldsymbol{r}', \boldsymbol{r}'' \rangle = 0$ and $\langle \boldsymbol{w}', \boldsymbol{r}'' \rangle + \langle \boldsymbol{r}', \boldsymbol{w}'' \rangle = 0$.

In other words, we pick randomness which does not induce any errors. Thus, we do not need to send anything besides $[c_{\boldsymbol{r}}] = [\boldsymbol{g}']\boldsymbol{r}' + [\boldsymbol{g}'']\boldsymbol{r}''$ to the verifier. Unfortunately, due to the constraints dependency on the witness, the protocol is not zero-knowledge in the usual sense, but only, what we call, *almost* statistical zero-knowledge.

Now, we outline our almost zero-knowledge argument, using an augmented masking set $\mathbb{M}_n^+$ which is defined later.

Protocol 4.4.3 ($\mathsf{IPA}_{\text{almZK}}$). The following is an inner product argument for $\mathcal{R}_{\mathsf{IPA}}$ as in Protocol 4.4.1 ($\mathsf{IPA}_{\text{noZK}}$).

- $\mathsf{P} \to \mathsf{V}$: Pick $\boldsymbol{r}' \xleftarrow{\$} \mathbb{M}_n^+$. Pick $\boldsymbol{r}''$ uniformly from $\mathbb{M}_n^+$ subject to $\langle \boldsymbol{r}', \boldsymbol{r}'' \rangle = 0$ and $\langle \boldsymbol{w}', \boldsymbol{r}'' \rangle = -\langle \boldsymbol{r}', \boldsymbol{w}'' \rangle$, that is, uniformly from $\{\boldsymbol{v} \in \mathbb{M}_n^+ \mid \langle \boldsymbol{r}', \boldsymbol{r}'' \rangle = 0 \wedge \langle \boldsymbol{w}', \boldsymbol{r}'' \rangle = -\langle \boldsymbol{r}', \boldsymbol{w}'' \rangle\}$. Compute $[c_{\boldsymbol{r}}] := [\boldsymbol{g}']\boldsymbol{r}' + [\boldsymbol{g}'']\boldsymbol{r}''$. Send $[c_{\boldsymbol{r}}]$.
- $\mathsf{V} \to \mathsf{P}$: Pick and send $\beta \xleftarrow{\$} \chi^{(\beta)}$.
- $\mathsf{P} \leftrightarrow \mathsf{V}$: Engage in Protocol $\mathsf{IPA}_{\text{noZK}}$ for $\langle \beta\boldsymbol{w}' + \boldsymbol{r}', \beta\boldsymbol{w}'' + \boldsymbol{r}'' \rangle = \beta^2 t$ (with commitment $[c] = [c_{\boldsymbol{r}}] + \beta[c_{\boldsymbol{w}}] + \beta^2 t[Q]$). Verifier (and prover) use t (and $[c_{\boldsymbol{w}}]$) from the statement to compute $[c]$.

See Appendix B.7 for a sketch of this protocol.

Correctness follows by inspection. Soundness follows essentially from Lemma 4.4.2 and Lemma 4.3.2.

Corollary 4.4.4. *Protocol 4.4.3 is $(2, 2, 2k, \ldots, 2k)$-special (relaxed) sound for finding a witness for $\mathcal{R}_{\mathsf{IPA}}$ or a non-trivial element in the kernel of $[\boldsymbol{g}', \boldsymbol{g}'', Q]$. Moreover, it has $(2, 1, k, \ldots, k)$-quick $(2, 2, 2k, \ldots, 2k)$-short extractability.*

Showing zero-knowledge is more contrived. As for $\mathsf{LMPA}_{\text{ZK}}$ in Lemma B.4.5, we want to show that the prover's messages are uniformly random. Unfortunately, the constraints which must be satisfied depend on the witness. Thus, an adversarially chosen witness may be a problem. Hence, we require certain properties, which automatically hold when $\mathsf{IPA}_{\text{almZK}}$ is used with suitably "randomised" witnesses, e.g. commitments.

Definition 4.4.5. Let k be fixed and $n \geq 4k$. Define $\mathbb{M}_n^+ := \mathbb{M}_n \dot{\cup} \{n/k + 1, n/k + 2\}$.

We introduce $\mathbb{M}_n^+$ because satisfying the kernel constraints "consumes two pieces of randomness" in $\boldsymbol{r}''$. We compensate this in $\mathbb{M}_n^+$. Note that for this, $\{n/k + 1, n/k + 2\}$ must be disjoint from $\mathbb{M}_n$, which is guaranteed by $n \geq 4k$. For the sake of simplicity, we will from now on stick to $k = 2$. But it should be evident how to appropriately generalise, cf. Appendix B.4.

Lemma 4.4.6. *Let crs $= ([\boldsymbol{g}', \boldsymbol{g}'', Q])$ be uniformly random and as in Protocol 4.4.3 ($\mathsf{IPA}_{\text{almZK}}$) and $k = 2$. Suppose that $n \geq 4k$ and let $\mathbb{M}_n^+$ be as in Definition 4.4.5. Suppose $\boldsymbol{w}'$ is chosen such that some component of w_1', w_2', w_{n-1}', or w_n' is uniformly random independent of $[\boldsymbol{g}'']$. Then $\mathsf{IPA}_{\text{almZK}}$ is ε-statistical HVZK with $\varepsilon = 2/p + 2(k-1)\log_2(n)/p$ for such witness distributions.*

More generally, we have the following: For arbitrary but fixed $\boldsymbol{w}'$, $\boldsymbol{w}''$, x_1, x_2, consider the combined transition and constraint matrix

$$\underbrace{\begin{pmatrix} x_1 \operatorname{id}_{n/2} & x_2 \operatorname{id}_{n/2} \\ \boldsymbol{g}_2'' & 0 \\ 0 & \boldsymbol{g}_1'' \\ \hline \boldsymbol{w}_1'^\top & \boldsymbol{w}_2'^\top \\ \boldsymbol{r}_1'^\top & \boldsymbol{r}_2'^\top \end{pmatrix}}_{=:\widetilde{\boldsymbol{M}}''} \begin{pmatrix} \boldsymbol{r}_1'' \\ \boldsymbol{r}_2'' \end{pmatrix} = \begin{pmatrix} \widehat{\boldsymbol{r}}' \\ u_{-1}'' \\ u_{+1}'' \\ \hline -\langle \boldsymbol{r}', \boldsymbol{w}'' \rangle \\ 0 \end{pmatrix}.$$

where $\boldsymbol{r}' \xleftarrow{\$} \mathbb{M}_n^+$. Let $\boldsymbol{M}''$ be defined by $\widetilde{\boldsymbol{M}}''$ but restricted to non-zero components of $\boldsymbol{r}''$ resp. $\widehat{\boldsymbol{r}}''$, i.e. to columns in $\mathbb{M}_n^+$ resp. in to rows in $\mathbb{M}_{n/2}$ in upper block of $\widetilde{\boldsymbol{M}}''$. Let $\boldsymbol{C}''$ be $\boldsymbol{M}''$ except that the last row $(\boldsymbol{r}_1'^\top, \boldsymbol{r}^\top)$ of $\boldsymbol{M}''$ is removed. Let $\mathcal{A}$ be an unbounded HVZK adversary which picks the witnesses $(\boldsymbol{w}', \boldsymbol{w}'')$ (given $([\boldsymbol{g}', \boldsymbol{g}'', Q])$). Let δ be an upper bound on the probability that $\mathcal{A}$ chooses $(\boldsymbol{w}', \boldsymbol{w}'')$ such that $\boldsymbol{C}''$ is not surjective. Then the advantage against HVZK of $\mathcal{A}$ is at most $\delta + (1 + 2(k-1)\log(n))/p$.

The idea and proof of Lemma 4.4.6 is very similar to Lemma 4.3.23, and we argue via surjectivity of transition matrices The main difference is the first recursive step, where $\mathbb{M}_n^+$ and the constraints on $\boldsymbol{r}''$ play a role. After that, the argument of Lemma 4.3.23 applies directly since the transition matrix for $\boldsymbol{r}''$ (and $u_{\pm 1}''$) are identical to those in Lemma 4.3.23. Generalisations of this result to $k \geq 2$ are left to the reader. For simplicity, in the proof, we exploit that $\boldsymbol{r}''$ alone is sufficient to ensure all $[u_\ell]$ are randomised (until $n = k$), and then we use that $\mathbb{M}_k = \{1, \dots, k\}$, i.e. $\boldsymbol{r}'$ resp. $\boldsymbol{r}''$ completely mask the witness at that point. The full proof is in Appendix B.3

4.4.5. Quadratic Equation Satisfiability

We can finally instantiate our sketch of an argument system for satisfiability of a system of quadratic equations from Section 4.4.3. It is a commit-and-prove system as follows. The prover commits to the solution $\boldsymbol{w}$. Then $\boldsymbol{\Gamma}$ is fixed and $\langle \boldsymbol{w}, \boldsymbol{\Gamma w} \rangle = 0$ shown to hold. The commitment scheme pads $\boldsymbol{w} \in \mathbb{F}_p^{n-2}$ with randomness and extends $\boldsymbol{\Gamma}$ in a suitable way. Intuition for soundness is given in Section 4.4.3.

Protocol 4.4.7 (QESA$_{\mathrm{ZK}}$). Let $\{\boldsymbol{\Gamma}_i \in \mathbb{F}_p^{(n-2)\times(n-2)} \mid i = 1, \dots, N\}$ be a system of quadratic equations. Suppose $N \geq 2$. Let $\boldsymbol{w} \in \mathbb{F}_p^{n-2}$ be a solution, i.e. $\boldsymbol{w}^\top \boldsymbol{\Gamma}_i \boldsymbol{w} = 0$ for all i. We assume that the first component w_1 of $\boldsymbol{w}$ is 1.

Let $crs = [\boldsymbol{g}', \boldsymbol{g}'', Q]$, $\widetilde{\chi}_{2k-1}$, $\chi^{(\beta \neq 0)}$ and $n \geq 4k$ as in Protocol 4.4.3, and $\mathbb{M}_n^+$ as in Lemma 4.4.6. Let $\boldsymbol{x} \leftarrow \chi_N$ be a testing distribution with $x_1 = 1$ and $x_2 \neq 0$ for all $\boldsymbol{x}$.[19] Let $\boldsymbol{y} \leftarrow \chi_{n+1}$ be a testing distribution with $y_1 = 1$ always. The following is a protocol for proving

$$\exists \boldsymbol{w} \in \mathbb{F}_p^{n-2}: \quad \forall i: \boldsymbol{w}^\top \boldsymbol{\Gamma}_i \boldsymbol{w} = 0 \ \wedge \ w_1 = 1$$

where crs and $\{\boldsymbol{\Gamma}_i\}_i$ are common inputs and the prover's witness is $\boldsymbol{w}$ satisfying $\{\boldsymbol{\Gamma}_i\}$ and $w_1 = 1$.

1. P $\rightarrow$ V (Commitment to $\boldsymbol{w}'$): Pick $\boldsymbol{r}' \xleftarrow{\$} \mathbb{F}_p^2$. Let the "extended" witness be $\boldsymbol{w}' := \left(\begin{smallmatrix} \boldsymbol{w} \\ \boldsymbol{r}' \end{smallmatrix}\right)$ and compute the commitment $[c_w'] = [\boldsymbol{g}']\boldsymbol{w}'$. Send $[c_w']$.
2. V $\rightarrow$ P (Batch equations and fix w_1 to 1):

[19] Restrictions on χ_N are merely to simplify protocol description and proofs.

- Pick and send $\boldsymbol{x} \xleftarrow{\$} \chi_N$. Both parties compute $\boldsymbol{\Gamma} := \sum x_i \boldsymbol{\Gamma}_i \in \mathbb{F}_p^{(n-2)\times(n-2)}$.
- Both parties let $\beta := x_2$ and do: Redefine $[g'_1] := \beta^{-1}[g'_1]$. Redefine $[c'_{\boldsymbol{w}}] \leftarrow [c'_{\boldsymbol{w}}] - (\beta - 1)[g'_1]$ (with the new $[g'_1]$).

3. P $\longrightarrow$ V: Let $\boldsymbol{r}'' = \boldsymbol{R}\boldsymbol{r}'$ where $\boldsymbol{R} = \left(\begin{smallmatrix} 0 & -1 \\ 1 & 0 \end{smallmatrix}\right)$ is a rotation by 90 degrees. Let $\boldsymbol{w}'' = \left(\begin{smallmatrix} \boldsymbol{\Gamma}\boldsymbol{w} \\ \boldsymbol{r}'' \end{smallmatrix}\right)$. Compute and send $[c''_{\boldsymbol{w}}] = [\boldsymbol{g}'']\boldsymbol{w}''$.
4. V $\longrightarrow$ P: Pick $(1, \bar{s}, \underline{s})^\top \xleftarrow{\$} \chi_{n+1}$ where $\bar{s} \in \mathbb{F}_p^{n-2}$, $\underline{s} \in \mathbb{F}_p^2$. Let $\boldsymbol{s}' = (\bar{s}, \underline{s})^\top$ and send $\boldsymbol{s}'$.
5. P $\leftrightarrow$ V: Both parties let $\boldsymbol{\Gamma}' = \left(\begin{smallmatrix} \boldsymbol{\Gamma} & 0 \\ 0 & \boldsymbol{R} \end{smallmatrix}\right) \in \mathbb{F}_p^{n\times n}$ where $\boldsymbol{R}$ is as in Step 1. Then they engage in Protocol $\mathsf{IPA}_{\mathrm{almZK}}$ for $\langle \boldsymbol{w}' - \boldsymbol{s}', \boldsymbol{w}'' + \boldsymbol{\Gamma}'^\top \boldsymbol{s}' \rangle = t$ with $t = -\langle \underline{s}, \boldsymbol{\Gamma}^\top \underline{s} \rangle$, and commitment $[c] = ([c'_{\boldsymbol{w}}] - [\boldsymbol{g}']\boldsymbol{s}') + ([c''_{\boldsymbol{w}}] + [\boldsymbol{g}'']\boldsymbol{\Gamma}'^\top \boldsymbol{s}')$ and the modified $[\boldsymbol{g}']$ (and unmodified $[\boldsymbol{g}'']$, $[Q]$) as commitment keys.

See Appendix B.7 for a sketch of this protocol.

Remark 4.4.8. It is not hard to see that the prover never needs to compute $[c] = ([c'_{\boldsymbol{w}}] - [\boldsymbol{g}']\boldsymbol{s}') + ([c''_{\boldsymbol{w}}] + [\boldsymbol{g}'']\boldsymbol{\Gamma}'^\top \boldsymbol{s}')$. (In general, P does not need $[\boldsymbol{u}_0]$.) While the verifier has to check $[c]$, using lazy evaluation and optimisations from [BBB+18], this hardly affects its runtime. All in all, dealing with $\boldsymbol{s}'$ is almost free.

Remark 4.4.9. Compared to the idea derived in Section 4.4.3, the challenge α does not appear in Protocol 4.4.7. The reason is simple: Every appearance of $\alpha\boldsymbol{s}$ in Section 4.4.3 corresponds to an appearance of $\boldsymbol{s}'$ in Protocol 4.4.7, i.e. α is absorbed into $\boldsymbol{s}$, and thus turns out to be superfluous.

Lemma 4.4.10. *Protocol* $\mathsf{QESA}_{\mathsf{ZK}}$ *has perfect correctness.*

Using $\langle \left(\begin{smallmatrix} \boldsymbol{u}' \\ \boldsymbol{r}' \end{smallmatrix}\right), \left(\begin{smallmatrix} \boldsymbol{u}'' \\ \boldsymbol{r}'' \end{smallmatrix}\right) \rangle = \langle \boldsymbol{u}', \boldsymbol{u}'' \rangle + \langle \boldsymbol{r}', \boldsymbol{r}'' \rangle$ and $\langle \boldsymbol{r}, \boldsymbol{R}\boldsymbol{r} \rangle = 0$ for all $\boldsymbol{r} \in \mathbb{F}_p^2$, correctness follows by a straightforward check.

Proof. The verifier only rejects an honest prover if the $\mathsf{IPA}_{\mathrm{almZK}}$ rejects, and $\mathsf{IPA}_{\mathrm{almZK}}$ is perfectly correct. Hence, it suffices to show that the statment which $\mathsf{IPA}_{\mathrm{almZK}}$ proves, i.e. $\langle \boldsymbol{w}' - \boldsymbol{s}', \boldsymbol{w}'' + \boldsymbol{\Gamma}'^\top \boldsymbol{s}' \rangle = t$, for $t = -\langle \underline{s}, \boldsymbol{\Gamma}^\top \underline{s} \rangle$, always holds (for honest prover). Let $\boldsymbol{w}' = \left(\begin{smallmatrix} \boldsymbol{w} \\ \boldsymbol{r}' \end{smallmatrix}\right)$, $\boldsymbol{w}'' = \left(\begin{smallmatrix} \boldsymbol{\Gamma}\boldsymbol{w} \\ \boldsymbol{R}\boldsymbol{r}' \end{smallmatrix}\right)$ as in the protocol and let $\boldsymbol{s}' = \begin{pmatrix} \bar{s} \\ \underline{s} \end{pmatrix}$ with $\bar{s}$, $\underline{s}$ as in $\mathsf{QESA}_{\mathsf{ZK}}$. By construction, $\langle \boldsymbol{x}, \boldsymbol{R}\boldsymbol{x} \rangle = 0$ for any $\boldsymbol{x} \in \mathbb{F}_p^2$. With this, we find

$$\begin{aligned} \langle \boldsymbol{w}' - \boldsymbol{s}', \boldsymbol{w}'' + \boldsymbol{\Gamma}'^\top \boldsymbol{s}' \rangle &= \langle \boldsymbol{w} - \bar{s}, \boldsymbol{\Gamma}\boldsymbol{w} + \boldsymbol{\Gamma}^\top \bar{s} \rangle + \langle \boldsymbol{r}' - \underline{s}, \boldsymbol{R}\boldsymbol{r}' + \boldsymbol{R}^\top \underline{s} \rangle \\ &= -\langle \underline{s}, \boldsymbol{\Gamma}^\top \underline{s} \rangle \end{aligned}$$

because from $\boldsymbol{R}^\top = -\boldsymbol{R}$ and $\langle \boldsymbol{w}, \boldsymbol{\Gamma}\boldsymbol{w} \rangle = 0$ we have

$$\begin{aligned} \langle \boldsymbol{r}' - \underline{s}, \boldsymbol{R}\boldsymbol{r}' + \boldsymbol{R}^\top \underline{s} \rangle &= \langle \boldsymbol{r}' - \underline{s}, \boldsymbol{R}(\boldsymbol{r}' - \underline{s}) \rangle = 0 \\ \text{and} \quad \langle \boldsymbol{w} - \bar{s}, \boldsymbol{\Gamma}\boldsymbol{w} + \boldsymbol{\Gamma}^\top \bar{s} \rangle &= \underbrace{\langle \boldsymbol{w}, \boldsymbol{\Gamma}\boldsymbol{w} \rangle}_{=0} + \underbrace{(\langle \boldsymbol{w}, \boldsymbol{\Gamma}^\top \bar{s} \rangle - \langle \bar{s}, \boldsymbol{\Gamma}\boldsymbol{w} \rangle)}_{=0} - \langle \underline{s}, \boldsymbol{\Gamma}^\top \underline{s} \rangle \end{aligned}$$

and thus

$$\langle \boldsymbol{w}' - \boldsymbol{s}', \boldsymbol{w}'' + \boldsymbol{\Gamma}'^\top \boldsymbol{s}' \rangle = -\langle \underline{s}, \boldsymbol{\Gamma}^\top \underline{s} \rangle. \qquad \square$$

Lemma 4.4.11. *Protocol* $\mathsf{QESA}_{\mathsf{ZK}}$ *is* $(N, n+1, 2, 2, 2k, \dots, 2k)$*-special sound for relaxed soundness relation of finding a witness in* $\mathcal{R}_{\mathsf{IPA}}$ *or a non-trivial kernel element of* $[\boldsymbol{g}', \boldsymbol{g}'', Q]$. *Moreover, it inherits* $(1, 1, 2, 2, k, \dots, k)$*-quick* $(N, n+1, 2, 2, 2k, \dots, 2k)$*-short extractability.*

Proof. First of all, note that the randomisation of $[g'_1]$ to fix w_1 to 1 is as in Lemma 4.4.2. In particular, non-trivial kernel elements in $[\boldsymbol{g}'_{\text{new}}]$ yield such in $[\boldsymbol{g}'_{\text{old}}]$. Therefore, it suffices to only consider $[\boldsymbol{g}'] = [\boldsymbol{g}'_{\text{new}}]$ in the following.

Arguing the claimed extractability is straightforward. In the subprotocol of $\mathsf{IPA}_{\text{almZK}}$, we have the claimed short-circuit extraction by Corollary 4.4.4. From this, we (dynamically) find for each the i-th subtree witnesses $\boldsymbol{w}'_i, \boldsymbol{w}''_i, t_i$ with $[\boldsymbol{g}', \boldsymbol{g}'', Q]\begin{pmatrix} \boldsymbol{w}'_i \\ \boldsymbol{w}''_i \\ t_i \end{pmatrix}$. If these are not kernel elements of $[\boldsymbol{g}', \boldsymbol{g}'', Q]$, they satisfy $\mathcal{R}_{\mathsf{IPA}}$, and short-circuit-extraction holds. If we ever encounter a $(\boldsymbol{w}'_i, \boldsymbol{w}''_i, t_i) \neq (\boldsymbol{w}'_j, \boldsymbol{w}''_j, t_j)$ for $i \neq j$, this yields a non-trivial kernel element, hence short-circuits. Thus, in the following, w.l.o.g. we assume that $\boldsymbol{w}' = \boldsymbol{w}'_i$, $\boldsymbol{w}'' = \boldsymbol{w}''_i$, $t = t_i = \langle \underline{\boldsymbol{s}}, \boldsymbol{\Gamma}^\top \underline{\boldsymbol{s}}_i \rangle$ for all i.

We're left with a $(N, n+1)$-tree of transcripts. We first deal with an $(n+1)$-subtree of the penultimate challenge. We have with following properties: The witness on each node in (layer 2) is the same fixed $(\boldsymbol{w}', \boldsymbol{w}'')$ which satisfies

$$\langle \boldsymbol{w}' - \boldsymbol{s}', \boldsymbol{w}'' + \boldsymbol{\Gamma}'^\top \boldsymbol{s}' \rangle = -\langle \underline{\boldsymbol{s}}, \boldsymbol{\Gamma}^\top \underline{\boldsymbol{s}} \rangle$$

where $\boldsymbol{s}' = \begin{pmatrix} \bar{\boldsymbol{s}} \\ \underline{\boldsymbol{s}} \end{pmatrix}$ denotes the respective challenge. We find

$$\begin{aligned} 0 &= \langle \boldsymbol{w}' - \boldsymbol{s}', \boldsymbol{w}'' + \boldsymbol{\Gamma}'^\top \boldsymbol{s}' \rangle + \langle \boldsymbol{s}', \boldsymbol{\Gamma}^\top \boldsymbol{s}' \rangle \\ &= \langle \boldsymbol{w}', \boldsymbol{w}'' \rangle + \langle \boldsymbol{\Gamma}' \boldsymbol{w}' - \boldsymbol{w}'', \boldsymbol{s}' \rangle \\ &= (1, \boldsymbol{s}'^\top) \begin{pmatrix} \langle \boldsymbol{w}', \boldsymbol{w}'' \rangle \\ \boldsymbol{\Gamma}' \boldsymbol{w}' - \boldsymbol{w}'' \end{pmatrix} \end{aligned}$$

Given $n+1$ accepting transcripts for linearly independent challenges $\left(\begin{smallmatrix} 1 \\ \boldsymbol{s}' \end{smallmatrix}\right)$, we find that $\langle \boldsymbol{w}', \boldsymbol{w}'' \rangle = 0$ and $\boldsymbol{\Gamma}' \boldsymbol{w}' - \boldsymbol{w}'' = \boldsymbol{0}$. Now, let $\boldsymbol{w}' = \left(\begin{smallmatrix} \boldsymbol{u}' \\ \boldsymbol{r}' \end{smallmatrix}\right)$ with $\boldsymbol{u}'$ in $\mathbb{F}_p^{n-2}$ and $\boldsymbol{r}' \in \mathbb{F}_p^2$, and likewise $\boldsymbol{w}'' = \left(\begin{smallmatrix} \boldsymbol{u}'' \\ \boldsymbol{r}'' \end{smallmatrix}\right)$. We find (using the block structure of $\boldsymbol{\Gamma}'$) that

$$\boldsymbol{r}'' = \boldsymbol{R} \boldsymbol{r}' \quad \text{and} \quad \boldsymbol{u}'' = \boldsymbol{\Gamma} \boldsymbol{u}'.$$

Thus, $\langle \boldsymbol{r}', \boldsymbol{r}'' \rangle$ and we get

$$0 = \langle \boldsymbol{w}', \boldsymbol{w}'' \rangle = \langle \boldsymbol{u}', \boldsymbol{u}'' \rangle + \underbrace{\langle \boldsymbol{r}', \boldsymbol{r}'' \rangle}_{=0}.$$

Consequently, $\boldsymbol{w} := \boldsymbol{u}'$ is a witness with $\boldsymbol{w}^\top \boldsymbol{\Gamma} \boldsymbol{w} = 0$.

What's left to show is that given N transcripts (with linearly independent challenges), we must have $w_1 = 1$ and a solution to $\{\boldsymbol{\Gamma}_i\}_i$ (or have found a non-trivial kernel element) The redefinition of $[g'_1]$ is equivalent to setting w_1 to $w_1 := \beta w_1 - (\beta - 1)$. Since, like $\boldsymbol{w}'_{\text{new}}$, all $\boldsymbol{w}'_{\text{old}}$ must coincide (or a non-trivial kernel element of $[\boldsymbol{g}'_{\text{new}}]$ is found, leading to short-circuit extraction). But clearly, under this condition we get $w_1 = 1$ if any pair $\beta \neq \beta'$ is in the transcripts. And since all N challenges $\boldsymbol{x}$ are linearly independent and $\beta = x_2$, this must be the case.

Finally, we prove that $\boldsymbol{w}$ satisfies $\{\boldsymbol{\Gamma}_i\}_i$. (Again, we assume that $\boldsymbol{w}$ is the same fixed one for all leafs, otherwise, short-circuit extractions would give a non-trival kernel element already.) Consider the vector $\boldsymbol{e} \in \mathbb{F}_p^N$ defined by $e_j := \boldsymbol{w}^\top \boldsymbol{\Gamma}_j \boldsymbol{w}$. Write $\boldsymbol{\Gamma}^{(i)} := \sum_j x_j^{(i)} \boldsymbol{\Gamma}_j$, where the superscript i indicates the i-th transcript. Since all transcripts are valid, we know that $\boldsymbol{w}^\top \boldsymbol{\Gamma}^{(i)} \boldsymbol{w} = 0$. We get

$$0 = \boldsymbol{w}^\top \boldsymbol{\Gamma}^{(i)} \boldsymbol{w} = \sum_j x_j^{(i)} \boldsymbol{w}^\top \boldsymbol{\Gamma}_j \boldsymbol{w} = \boldsymbol{x}^{(i)} \boldsymbol{e}.$$

Since all $\boldsymbol{x}^{(i)}$ are linearly independent this implies $\boldsymbol{e} = 0$, i.e. $\boldsymbol{w}$ solves each equation. □

If we use tensor-based testing distributions, as in Section 4.3.3.2 to improve soundness, we obtain following stronger guarantee.

Corollary 4.4.12. *If the testing distributions χ_N resp. χ_{n+1} are instantiated with a tensor-based distributions (cf. Example 4.2.12 and Section 4.3.3.2), then we achieve $((1,\dots,1),(1,\dots,1),2,2,k,\dots,k)$-quick and $((2,\dots,2),(2,\dots,2),2,2,2k,\dots,2k)$-short extractability.*

In particular, suppose that $k = 2$ (or $k = \mathcal{O}(1)$ more generally) and n is a power of 2 and $N \approx n$. Then by Corollary 4.2.19 the maximal size of a short-circuit tree is $\approx 8n\log(n)$ (resp. $\mathcal{O}(n\log_2(n))$).

Corollary 4.4.12 follows from Lemma 4.4.11 by viewing tensor-based testing as multiple challenges, completely analogous to the proof in Corollary 4.3.11.

Lemma 4.4.13. *Protocol $\mathrm{QESA}_{\mathrm{ZK}}$ is ε-statistical HVZK for $\varepsilon = 2/p + 2\log_2(n)/p$.*

For the proof of Lemma 4.4.13, we essentially establish that the conditions of Lemma 4.4.6 are met. Thus, $\mathsf{IPA}_{\mathrm{almZK}}$ is statistical HVZK, and consequently $\mathrm{QESA}_{\mathrm{ZK}}$ is statistical HVZK as well.

Proof. First, observe that simulation by computation in reverse works for all steps before the subprotocol call to $\mathsf{IPA}_{\mathrm{almZK}}$. Namely, the honest distributions of $[c'_{\boldsymbol{w}}]$ and $[c''_{\boldsymbol{w}}]$ are jointly uniform, and given two out of $[c'_{\boldsymbol{w}}]$, $[c''_{\boldsymbol{w}}]$, $[c]$, the third is uniquely defined. Indeed, the simulator can even choose $[c'_{\boldsymbol{w}}]$ and $[c''_{\boldsymbol{w}}]$ uniformly and compute $[c]$ appropriately. Thus, this part of the simulation is perfect. Hence, it suffices to show that the conditions of Lemma 4.4.6 are met to obtain the promised ε-statistical simulation. For this, observe that $\boldsymbol{w}' = \binom{\boldsymbol{w}}{\boldsymbol{r}'}$ chooses $\boldsymbol{r}'$ uniformly and independent of $[\boldsymbol{g}', \boldsymbol{g}'', Q]$. Thus, component n (and $n-1$) of $\boldsymbol{w}'$ satisfies the requirement of Lemma 4.4.6 and the claim follows. □

4.4.6. Combining $\mathrm{QESA}_{\mathrm{ZK}}$ with Other Proof Systems

As is, $\mathrm{QESA}_{\mathrm{ZK}}$ can be used to commit-and-prove quadratic equations. However, oftentimes, one wishes to prove statements about commitments which come from some other source (and do not include auxiliary information necessary for a proof). For example, Bulletproofs [BBB+18] were specifically designed for confidential transactions, where the commitments are *input* to the proof system. This is not immediately feasible with $\mathrm{QESA}_{\mathrm{ZK}}$ as is, because $\mathrm{QESA}_{\mathrm{ZK}}$ is commit-and-prove only w.r.t. the solution of the set of quadratic equations. So either the commitment *includes* the auxiliary information (e.g. a bit decomposition for range proofs), or $\mathrm{QESA}_{\mathrm{ZK}}$ is not directly applicable. Fortunately, applying (the ideas of) $\mathrm{QESA}_{\mathrm{ZK}}$ in such circumstances is not hard.

We consider following setting. There are commitment keys $\widetilde{ck}^{(i)}$ for $i = 1,\dots,M$. Each commitment key corresponds to a subset $\mathcal{I}_i \subseteq \{1,\dots,n\}$ of the components of $[\boldsymbol{g}']$, where $crs = ([\boldsymbol{g}', \boldsymbol{g}'', Q])$ is the commitment key of $\mathrm{QESA}_{\mathrm{ZK}}$. That is $\widetilde{ck}^{(i)} \mathrel{\widehat{=}} \{[g'_j]\}_{j\in\mathcal{I}_i}$. Let $\mathcal{I} := \cup_{i=1}^{M}\mathcal{I}_i$ be the set of all indices which are part of some $\widetilde{ck}^{(i)}$. Note that $\#\mathcal{I}_i$ is the size of $\widetilde{ck}^{(i)}$. We assume the following: Every commitment key $\widetilde{ck}^{(i)}$ uses $[g'_n]$ (or $[g'_{n-1}]$) as its randomness components. Moreover, $1 \notin \mathcal{I}_i$, because the index $1 \mathrel{\widehat{=}} [g'_1]$ is reserved for the commitment to value 1 in $\mathrm{QESA}_{\mathrm{ZK}}$. A useful point of view is that $\widetilde{ck}^{(i)}$ is a commitment under $[\boldsymbol{g}'] \in \mathbb{G}^n$ to a vector $\boldsymbol{v}^{(i)} \in \mathbb{F}_p^n$ with

$$\forall i \in \{1,\dots,n\} \forall j \notin \mathcal{I}_i \colon v_j = 0. \tag{4.4.2}$$

We assume for simplicity that there is only one commitment per commitment key $\widetilde{ck}^{(i)}$. To model the case of multiple commitments $[c_1],\dots,[c_M]$ for one key, e.g. all commitments are under $\widetilde{ck} = \widetilde{ck}^{(1)}$, we simply duplicate $\widetilde{ck}$, i.e. we rewrite this as $[\widetilde{c}^{(i)}] = [c_i]$, $\widetilde{ck}^{(i)} = \widetilde{ck}$.

Example 4.4.14. In a typical range proof, with Pedersen committed value, we would have $\widetilde{ck}^{(1)} \mathrel{\widehat{=}} [g'_2, g'_n]$, where $M = 1$. We write $\widetilde{ck} := \widetilde{ck}^{(1)}$ for simplicity. This means $\mathcal{I} = \{2, n\}$.

Remark 4.4.15. Using the in n varying $[g'_n]$ in the commitment keys $\widetilde{ck}^{(i)}$ is problematic and inconvenient. Running the protocol for a smaller or larger instance, e.g. an instance of size $n/2$ or $2n$, leads to incompatible requirements for randomizer terms of the commitment keys $\widetilde{ck}^{(i)}$. A simple solution is to fix some (random) $[g'^{,\star}_{\mathrm{rnd1}}, g'^{,\star}_{\mathrm{rnd2}}]$ (as part of *crs*) and construct $[\boldsymbol{g}']$ so that $[g'_{n-1}, g'_n] = [g'^{,\star}_{\mathrm{rnd1}}, g'^{,\star}_{\mathrm{rnd2}}]$. Another solution is to permute the position of the randomness and reserve the fixed indices $2, 3$ for randomness (instead of $n-1, n$). Either approach fixes the group elements corresponding to the randomising term, solving the problem.

With this setup, we can extend QESA$_{\mathrm{ZK}}$ as follows: Given commitments $[\widetilde{c}^{(i)}]$ under keys $\widetilde{ck}^{(i)}$, prove that the values committed in $[\widetilde{c}^{(i)}]$ satisfy some set of quadratic equations. In other words, prove that the $[\widetilde{c}^{(i)}]$ satisfy some arithmetic circuit.

Example 4.4.16 (Aggregate range proof). Consider $[\widetilde{c}^{(j)}]$, $j = 1, \ldots, M$. We wish to prove that the values $\boldsymbol{v}^{(j)}$ committed in $[\widetilde{c}^{(j)}]$ all lie in the range $\{0, \ldots, 2^8 - 1\}$.

Unsurprisingly, our solution to the problem is probabilistic verification. On a high level, we proceed as follows: The commitments $[\widetilde{c}^{(i)}]$ are part of the statement. We start QESA$_{\mathrm{ZK}}$ as usual, the prover sends the commitment $[c'_{\boldsymbol{w}}]$ to the witness, where the components in $\mathcal{I}$ are zeroed (except for the randomness in $n-1, n$). Then the verifier sends a challenge $\boldsymbol{\alpha} \in \mathbb{F}_p^{M+1}$ with $\alpha_0 = 1$. Both sides compute the random linear combination $[c'_{\boldsymbol{w}}] := [c'_{\boldsymbol{w}}] + \sum_{i=1}^{M} \alpha_i [\widetilde{c}^{(i)}]$ as the new commitment. The prover adjusts his (extended) witness $\boldsymbol{w}' = \left(\begin{smallmatrix}\boldsymbol{w}\\ r'\end{smallmatrix}\right)$ to $\boldsymbol{w}' \leftarrow \alpha_0 \boldsymbol{w}' + \sum_i \alpha_i \boldsymbol{v}^{(i)}$. The statements, i.e. the matrices $\boldsymbol{\Gamma}_i$ are also adjusted, and additional "glue equations" are included.

For a single commitment, the strategy can be made to work as described. For multiple commitments, it depends on the statement. The problem is, that if $\mathcal{I}_i$ and $\mathcal{I}_j$ are overlapping for $i \neq j$ outside the randomness components $\{n-1, n\}$, then one cannot recover the values for both $\boldsymbol{v}^{(i)}$ and $\boldsymbol{v}^{(j)}$ from their linear combination. Hence, this must be corrected.

Our idea for general interoperability is as follows. The initial QESA$_{\mathrm{ZK}}$ witness $\boldsymbol{w}$ (commitment $c'_{\boldsymbol{w}}$) has all components in $\mathcal{I}$ zeroed (except for randomness $n-1$, n) and also contains *copies* of the committed $\boldsymbol{v}^{(i)}$. The actual equations, i.e. the $\boldsymbol{\Gamma}_i$, only refer to the copies and the components $\mathcal{I}$. As before, for verifier randomness $\boldsymbol{\alpha}$, we set $[c'_{\boldsymbol{w}}] \leftarrow [c'_{\boldsymbol{w}}] + \sum_i \alpha_i [\widetilde{c}^{(i)}]$, and obtain $\boldsymbol{w}' \leftarrow \boldsymbol{w}' + \sum_i \alpha_i \boldsymbol{v}^{(i)}$ as new extended witness. Note that all (extended) equations $\boldsymbol{w}' \boldsymbol{\Gamma}'^{\top}_i \boldsymbol{w}'$ still hold (for an honest prover). Now we add (linear) equations $\boldsymbol{\Gamma}^{(i)}_{\mathrm{copy}}$ to the statement, which we call *copy-equations* and which depend on the randomness α_i. These equations simply assert that, if we compute $\sum_i \alpha_i \boldsymbol{v}^{(i)}$ using the committed copies in $\boldsymbol{w}$, then this equals the values committed in components $\mathcal{I}$ (again excluding the randomness components $n-1$, n). In other words, we assert that the purported copies of $\boldsymbol{v}^{(i)}$ in witness $[c'_{\boldsymbol{w},\mathrm{old}}]$ were valid copies. Commitment randomness (in components $\{n-1, n\}$) is *not* copied, as it is not a relevant part of the committed value. It is simply accumulated, e.g. as $r_2^* = \alpha_0 r_2 + \alpha_1 r^{(1)} + \alpha_2 r^{(2)}$. This "copy-based" approach is simple and modular.

The formulaic description of QESA$_{\mathrm{Copy}}$ is arguably technical. However, the examples in Fig. 4.3 and Fig. 4.4 demonstrate that it is a simple concept.

The presented approach has one small downside: It does not in fact prove that Eq. (4.4.2) holds, i.e. it does not enforce that $\widetilde{c}^{(i)}$ is zero outside of $\mathcal{I}_i$. If necessary, this can be enforced via different means, cf. Remark 4.4.18 . In applications, this often follows from prior knowledge anyway.

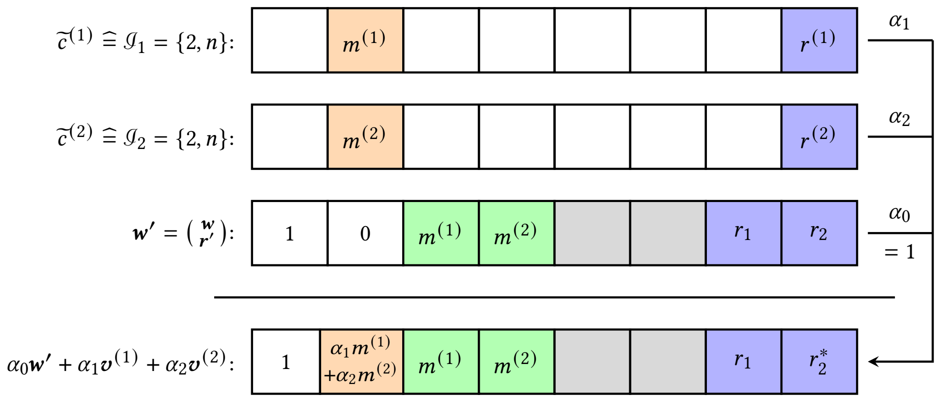


Figure 4.3.: An example of a copying two values from two commitments. The blocks are colour-coded as follows: White blocks contain either 0 or the value indicated. Orange blocks belong to the (value-part) of commitment indices, i.e. to $\mathcal{I}$. Green blocks denote "copied" values. Gray blocks contain an arbitrary value. Blue blocks refer to randomness parts (i.e. components $n-1, n$). Recall that randomness is *not* copied (i.e. components $\{n-1, n-2\}$). The actual statements (i.e. the matrices $\mathbf{\Gamma}_i$) are statements over all variables except the orange (and blue) blocks, as these are merely "test-values" which ensure that $\boldsymbol{w}$ contains copies of (the message part of) $\boldsymbol{v}^{(i)}$, here $m^{(i)}$, as claimed.

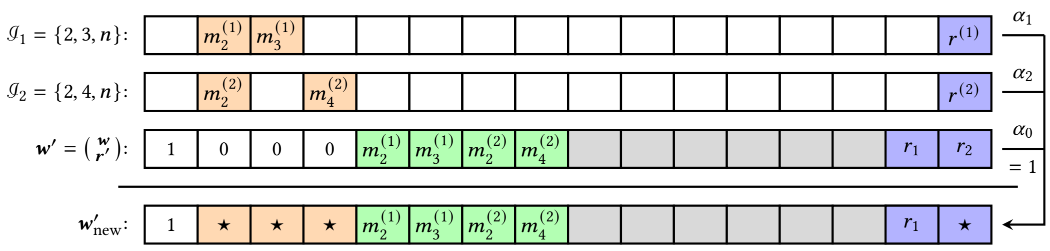


Figure 4.4.: This is a more complex example of the copying technique. Colour-coding is as before. Note that $\mathcal{I}_1 \neq \mathcal{I}_2$. Again, all orange values $\boldsymbol{m}^{(i)} \widehat{=} \boldsymbol{v}^{(i)}$, are copied and appear as green values in $\boldsymbol{w}$. Further optimisations are possible:

Protocol 4.4.17 ($\text{QESA}_{\text{Copy}}$). Let $n \geq 4k$, $\{\mathbf{\Gamma}_i\}_{i=1}^{N}$, $crs = [\boldsymbol{g}', \boldsymbol{g}'', Q]$, $\widetilde{\chi}_{2k-1}$ be as in Protocol QESA_{ZK}. Let χ_{M+1} be a testing distribution where the first component is always 1, i.e. $\boldsymbol{\alpha} \xleftarrow{\$} \chi_{M+1}$ has $\alpha_0 = 1$.[20] Let $\widetilde{ck}^{(i)} \widehat{=} \mathcal{I}_i$ be commitment keys for commitments (for $i = 1, \dots, M$), as described above. Let $[\widetilde{c}^{(i)}]$ be commitments to values $\boldsymbol{v}^{(i)}$. We identify $\boldsymbol{v}^{(i)}$ with a vector in $\mathbb{F}_p^n$ when necessary (satisfying Eq. (4.4.2)). Let $\boldsymbol{w} \in \mathbb{F}_p^{n-2}$ be a solution to $\{\mathbf{\Gamma}_i\}_i$, i.e. $\boldsymbol{w}^\top \mathbf{\Gamma}_i \boldsymbol{w} = 0$ for all i. We assume that $w_1 = 1$ and

$$\forall i \in \{1, \dots, M\}\ \forall j \in \mathcal{I}_i \cap \{2, \dots, n-2\}\colon\ w_j = 0. \tag{4.4.3}$$

We assume there is a map τ with $\tau(i, \cdot)\colon \{1, \dots, \#\mathcal{I}_i\} \to \{1, \dots, n\}$ such that

- $\forall i\colon \tau(i, n-1) = n-1 \wedge \tau(i, n) = n$
- $\forall i\, \forall j \in \mathcal{I}_i \setminus \{n-1, n\}\colon\ w_{\tau(i,j)} = \boldsymbol{v}_j^{(i)}$
- $\forall i\colon\ \tau(i, \cdot)$ is injective and the sets $\tau(i, \mathcal{I}_i \setminus \{n-1, n\})$ are disjoint for all i.

[20] The restriction $\alpha_0 = 1$ is just for convenience.

The map τ and its requirements encodes where components of each $\boldsymbol{v}^{(i)}$ are “stored in” $\boldsymbol{w}$ (ignoring randomness components $\{n-1, n\}$). Define the correctness relation $\mathcal{R}_{\mathrm{Qcopy}}$ as

$$\mathcal{R}_{\mathrm{Qcopy}} = \left\{ \boldsymbol{w} \in \mathbb{F}_p^{n-2}, \boldsymbol{v}^{(i)} \in \mathbb{F}_p^n \;\middle|\; \begin{array}{l} \forall i \in \{1,\dots,N\} \colon \boldsymbol{w}^\top \boldsymbol{\Gamma}_i \boldsymbol{w} = 0 \\ \wedge\, \forall i \in \{1,\dots,M\} \colon [\widetilde{c}^{(i)}] = [\boldsymbol{g}']\boldsymbol{v}^{(i)} \\ \wedge\, \forall i \in \{1,\dots,M\} \forall j \in \mathcal{I}_i \setminus \{n-1,n\} \colon w_{\tau(i,j)} = \boldsymbol{v}_j^{(i)} \\ \wedge\, \forall i \in \{1,\dots,M\} \forall j \notin \mathcal{I}_i \colon \boldsymbol{v}_j^{(i)} = 0 \end{array} \right\}$$

In particular, the prover’s witness consists of $\boldsymbol{w}$, $\boldsymbol{v}^{(1)}$, …, $\boldsymbol{v}^{(M)}$ as described above. The statement consists of $\{\boldsymbol{\Gamma}_j\}_{j=1}^N$ and $\{[\widetilde{c}^{(i)}]\}_{i=1}^M$.

- P ⟶ V: (Step 0: Commit to $\boldsymbol{w}$.) Send $[c'_{\boldsymbol{w}}] \coloneqq [\boldsymbol{g}']\boldsymbol{w}'$ with $\boldsymbol{w}' = \binom{\boldsymbol{w}}{\boldsymbol{r}'}$, where $\boldsymbol{w}$ is as outlined above and $\boldsymbol{r}' \xleftarrow{\$} \mathbb{F}_p^2$.
- V ⟶ P: (Step 1: Batch verification and statement adaption for QESA$_{\mathrm{ZK}}$) Pick and send $\boldsymbol{\alpha} \xleftarrow{\$} \chi_{M+1}$, where $(\alpha_0,\dots,\alpha_M) = \boldsymbol{\alpha} \in \mathbb{F}_p^{M+1}$ and where $\alpha_0 = 1$ always (by assumption). Both sides set $[c'_{\boldsymbol{w}}] \coloneqq [c'_{\boldsymbol{w}}] + \sum_{i=1}^M \alpha_i [\widetilde{c}^{(i)}]$. The prover sets $\boldsymbol{w}' \coloneqq \boldsymbol{w}' + \sum_{i=1}^M \alpha_i \boldsymbol{v}^{(i)} \in \mathbb{F}_p^n$. The set of equations is extended with additional equations, given by “copy-matrices” $\boldsymbol{\Gamma}_{\mathrm{copy}}^{(k)}$ for each $k \in \mathcal{I} \cap \{1,\dots,n-2\}$ as follows:
$$\boldsymbol{w}^\top \boldsymbol{\Gamma}_{\mathrm{copy}}^{(k)} \boldsymbol{w} = 0 \quad \widehat{=} \quad w_k = \sum_{i=1}^M \sum_{k \in \mathcal{I}_i \setminus \{n-1,n\}} \alpha_i w_{\tau(i,k)}.$$
These equations formalise that computing the (random) linear combination of the (purported) *copies* of $\boldsymbol{v}^{(i)}$ (as part of $\boldsymbol{w}'$) yield the same value as the (random) linear combination of the commitments, cf. Figs. 4.3 and 4.4. With these additional equations and the adapted witness, continue as in QESA$_{\mathrm{ZK}}$ (Step 2) without further changes.

See Appendix B.7 for a sketch of this protocol.

It not hard to see that QESA$_{\mathrm{Copy}}$ is correct. For zero-knowledge, we merely note that QESA$_{\mathrm{ZK}}$ is statistical HVZK, and by a completely analogous proof, QESA$_{\mathrm{Copy}}$ is statistical HVZK as well.

Remark 4.4.18. As noted before, QESA$_{\mathrm{Copy}}$ has relaxed soundness. Importantly, Eq. (4.4.3) is *not ensured* for the commitments $[\widetilde{c}^{(i)}]$ (let alone the stronger Eq. (4.4.2)). This is not a proof artefact, in fact, the honest prover strategy would convince a verifier even if all unused components in QESA$_{\mathrm{Copy}}$ (grey in Fig. 4.3) arbitrary values — nothing is proven them!

Nevertheless, Eq. (4.4.2) may hold in applications due to priori knowledge, or it may be enforced by (combinations of) other means, e.g. an additional (integrated) zero-knowledge proof of knowledge, or adding additional checks $w_j = 0$ to force all “unused” components to 0, or by specially deriving $[\boldsymbol{g}']$, e.g. via a dual-testing distribution. Establishing Eq. (4.4.2) when all $\boldsymbol{v}^{(i)}$ coincide, i.e. $\boldsymbol{v} = \boldsymbol{v}^{(i)}$ for all i, is typically easy and very efficient. As the details are simple but application-dependent, we leave them to the reader.

The relaxed soundness relation $\mathcal{R}_{\mathrm{Qcopy\text{-}relaxed}}$ of QESA$_{\mathrm{Copy}}$ is defined as

$$\mathcal{R}_{\mathrm{Qcopy\text{-}relaxed}} = \left\{ \boldsymbol{w} \in \mathbb{F}_p^{n-2}, \boldsymbol{v}^{(i)} \in \mathbb{F}_p^n \;\middle|\; \begin{array}{l} \forall i \in \{1,\dots,N\} \colon \boldsymbol{w}^\top \boldsymbol{\Gamma}_i \boldsymbol{w} = 0 \\ \wedge\, \forall i \in \{1,\dots,M\} \colon [\widetilde{c}^{(i)}] = [\boldsymbol{g}']\boldsymbol{v}^{(i)} \\ \wedge\, \forall i \in \{1,\dots,M\} \forall j \in \mathcal{I}_i \setminus \{n-1,n\} \colon w_{\tau(i,j)} = \boldsymbol{v}_j^{(i)} \end{array} \right\}$$

Lemma 4.4.19. *Let $(M+1, N', n+1, 2, 2, 2k, \dots, 2k)$. Then Protocol* $\mathsf{QESA}_{\mathsf{Copy}}$ *is μ-special (relaxed) sound for a witness for $\mathcal{R}_{\mathsf{Qcopy\text{-}relaxed}}$ or a non-trivial kernel element of $[\boldsymbol{g}', \boldsymbol{g}'', Q]$, where N' is the number of equations plus the number of copy equations. Moreover,* $\mathsf{QESA}_{\mathsf{Copy}}$ *is $(M+1, 1, 1, 2, 2, k, \dots, k)$-quick μ-short extractable.*

Proof sketch. The proof is straightforward, but the indexing is tedious. We only sketch it.

First of all, note that just like with $\mathsf{QESA}_{\mathsf{ZK}}$, we can for each run with randomness $\boldsymbol{\alpha}$ extract a witness $\overline{\boldsymbol{w}}'_{\boldsymbol{\alpha}}$ satisfying all $\boldsymbol{\Gamma}_i$, including additional copy-equations (which depend on $\boldsymbol{\alpha}$). To complete the extraction, we have to obtain the committed values $\boldsymbol{v}^{(i)}$ of $[\widetilde{c}^{(i)}]$ and $\boldsymbol{w}'$ of $[c_{\boldsymbol{w}}]$, and demonstrate their consistency of $\boldsymbol{v}^{(i)}$ with $\boldsymbol{w}'$.

First, we note that from $M+1$ linearly independent challenges $\boldsymbol{\alpha}$, we obtain openings $\boldsymbol{w}'$, $\boldsymbol{v}^{(i)}$ to each commitment under commitment key $[\boldsymbol{g}']$ in the usual way. Indeed, this is just (the extraction of) a batch proof of opening, see Section 4.3.4.

Now, we reinterpret the setting to avoid carrying around too many indices. The "copy proofs" essentially state the following: There is a "subvector" $(\boldsymbol{a}, \boldsymbol{b}_1, \dots, \boldsymbol{b}_M)$ of (a permutation of) $\boldsymbol{w}$ such that $\boldsymbol{b}_i$ consists of the copied values of all $\boldsymbol{v}^{(i)}$, and $\boldsymbol{a}$ should be zero. In $\overline{\boldsymbol{w}}'_{\boldsymbol{\alpha}}$, we have $\boldsymbol{a}_\alpha = \boldsymbol{a} + \sum_{i=1}^{M} \alpha_i \boldsymbol{v}^{(i)}$. Otherwise, or a non-trivial kernel element must be encountered upon extracting all $\boldsymbol{w}_\alpha$, which leads to short-circuit extraction. By the "copy equations" from Step 1, we also have[21]

$$\sum_{i=1}^{M} \alpha_i \boldsymbol{b}_i = \boldsymbol{a}_\alpha = \boldsymbol{a} + \sum_{i=1}^{M} \alpha_i \boldsymbol{v}^{(i)}.$$

Given $M+1$ linearly independent $\boldsymbol{\alpha}$ (and recall $\alpha_0 = 1$ by assumption), we find that $\boldsymbol{a} = 0$ and $\boldsymbol{b}_i = \boldsymbol{v}^{(i)}$. Thus, we find that the $\boldsymbol{v}^{(i)}$ satisfy the copy constraint (given by the map τ) in $\mathcal{R}_{\mathsf{Qcopy\text{-}relaxed}}$, and the constraint that $\boldsymbol{w}$ has zeroed all w_i with $i \in \mathcal{G}$ (which corresponds to $\boldsymbol{a}$). This completes the proof. $\square$

Remark 4.4.20. It is possible to optimize $\mathsf{QESA}_{\mathsf{Copy}}$ by eliminating unnecessary copies. For example, if a single commitment (i.e. $M = 1$) is "copied", the variables in $\mathcal{G} \setminus \{n-1, n\}$ are simply the α-fold of the actual value. By modifying the statement $\{\boldsymbol{\Gamma}_i\}_i$ appropriately, one can omit any "copy components" and "copy equations". This also works if $M > 1$, where one of the copies can be omitted (per component in $\mathcal{G}$).

Another optimization uses more general "gluing equations" than just "copying". Namely, if the values committed in $\widetilde{c}^{(i)}$ can be checked from other (auxiliary) values in $\boldsymbol{w}$ via quadratic equations, these equations can be used to "glue" committed values together. For example, in a bit-decomposition-based range proof for v, then by checking $v = \sum_{i=0}^{\ell-1} 2^i b_i$ one can evidently omit the copy of v in $\boldsymbol{w}$, since v can be recomputed by $\sum_{i=0}^{\ell-1} 2^i b_i$, and the $\{\boldsymbol{\Gamma}^{(k)}_{\mathrm{copy}}\}_k$ are easily adapted to work with the "virtual" copy $\sum_{i=0}^{\ell-1} 2^i b_i$ of v within $\boldsymbol{w}$.

Example 4.4.21. Consider the situation of (aggregate) range proofs in [BBB+18], that is, we have a commitment key $\widetilde{ck} := [g'_2, g'_n]$ and want to prove that commitments $[c_i]$, for $i = 1, \dots, M$, all under this key, contain values within a some ℓ-bit range. With $\mathsf{QESA}_{\mathsf{Copy}}$ and the optimization of Remark 4.4.20, the prover transmits $2\lceil \log(\ell M + 4) \rceil + 3$ group elements and 2 scalars. If $\ell M + 4$ is just below a power of 2, then this is slightly better than Bulletproofs(+) [BBB+18; CHJ+22]. However, as $\mathsf{QESA}_{\mathsf{Copy}}$ provides a weaker soundness guarantee (recall Remark 4.4.18), we have to compensate for that. A very simple way

[21] If we don't have this, $\mathsf{QESA}_{\mathsf{ZK}}$ extraction would short-circuit.

to fix this is to run a batch proof of knowledge of openings for all $[c_i']$ (cf. Section 4.3.4). The prover sends 1 group element and 2 scalars for this, resulting in $2\lceil\log(\ell M+4)\rceil+4$ group elements and 4 scalars, which is smaller than [BBB+18] but larger than [CHJ+22]. Alternatively, one may introduce additional equations to zero all "unused" components, as noted in Remark 4.4.18. Lastly, one may let the verifier choose all components in $[\boldsymbol{g}']$ except $[g_2']$ and $[g_n']$, e.g. via a dual testing distribution (which enforces $[c_i']$ has all components except for $[g_2']$ and $[g_n']$ zeroed). This should come with almost no overhead, but one has to redo the security analysis, both for (short-circuit) extractability and honest-verifier zero-knowledge.

4.5. Implementation

We implemented all protocols in C++17 using the RELIC toolkit [AG] for underlying group operations. Our instantiation uses $\mathbb{G}$ = Curve25519 and thus $\mathbb{F}_p = \mathbb{F}_{2^{255}-19}$. For a fair comparison, we implemented Bulletproofs on the same architecture with equal care. The code is available on GitHub.[22]

Remark 4.5.1. The implementation is based on a slightly different version of $\mathsf{IPA}_{\mathrm{almZK}}$, where *three* constraints $\langle \boldsymbol{r}', \boldsymbol{r}'' \rangle = \langle \boldsymbol{w}', \boldsymbol{r}'' \rangle = \langle \boldsymbol{r}', \boldsymbol{w}'' \rangle = 0$ are used. The revised version of $\mathsf{IPA}_{\mathrm{almZK}}$ has only *two* constraints (which simplifies security analysis). These changes do not affect performance.

Representing $\boldsymbol{\Gamma}$. All QESA protocols make use of sparse matrices $\boldsymbol{\Gamma}$. For efficient computation, a suitable representation is necessary. Decomposing $\boldsymbol{\Gamma}$ into a sum $\sum_i \boldsymbol{a}_i \boldsymbol{b}_i^\top$, similar to R1CS, allows for both runtime and memory optimisations. Note that vectors $\boldsymbol{a}_i$ and $\boldsymbol{b}_i$ are sparse themselves, allowing for even further optimisation via an appropriate data structure. For multiplications $\boldsymbol{\Gamma s}$, at most $m \sum_i k_i \ell_i$ scalar multiplications are necessary, where m, k_i, ℓ_i are the number of non-zero entries in $\boldsymbol{s}$, $\boldsymbol{a}_i$, $\boldsymbol{b}_i$. Thus, all operations remain polynomial in the input size.

Results. We benchmarked our protocols on an Intel Core i7-6600U CPU at 2.6GHz running Debian Stretch 4.9.168 using a single core. A point multiplication with a random 254-bit scalar takes on average 0.28ms on this platform. Table 4.3 shows how our aggregate range proofs $\mathsf{QESA}_{\mathrm{RP}}$ compare to Bulletproofs. For $\mathsf{QESA}_{\mathrm{RP}}$, the internal witness $\boldsymbol{w}$ contains 4 static elements: the constant 1, the aggregate element for $\mathsf{QESA}_{\mathrm{Copy}}$, and the 2 random elements added by $\mathsf{QESA}_{\mathrm{Inner}}$, cf. Appendix B.7. Hence, we select the range as a power of 2 minus 4, in order to keep the CRS size from expanding to the next power of two, which doubles the computation in our simple implementation. Our results show that $\mathsf{QESA}_{\mathrm{RP}}$ outperforms Bulletproofs for all tested parameters. Allowing batching randomnesses to be small further improves the performance (cf. $\mathsf{QESA}_{\mathrm{RP}}$ (short) for 140-bit random values). Note that the execution times given in [BBB+18] are lower, since a highly optimised library dedicated to a single elliptic curve was used instead of a general purpose library as in this work. However, since both protocols were benchmarked on the same platform with the same underlying library, the values in Table 4.3 give a fair comparison.

Note that we have not applied special optimisations to the verification algorithms and therefore show verification times in gray. Using *delayed (batch) verification*, e.g. as in [BBB+18], significantly improves verifier performance. Optimised *verification* performance of Bulletproofs and our proof systems is

[22] `https://github.com/emsec/QESA_ZK`

Parameters	Bulletproofs		$\mathsf{QESA}_{\mathsf{RP}}$		$\mathsf{QESA}_{\mathsf{RP}}$ (short)	
	P	V	P	V	P	V
60 bit	0.26	0.17	0.16	0.07	0.15	0.06
60 bit × 2	0.47	0.29	0.32	0.15	0.30	0.10
60 bit × 32	7.4	4.5	5.1	2.4	4.6	1.7
60 bit × 128	28.9	17.9	20.6	9.4	18.4	6.7
60 bit × 512	116	78.7	82.3	37.5	73.8	27.1
124 bit	0.46	0.29	0.32	0.15	0.29	0.11
124 bit × 32	14.9	9.2	10.4	4.7	9.3	3.4
124 bit × 128	59.7	36.8	41.4	18.9	37.2	13.5
124 bit × 512	238	147	165	75.4	149	54.6
252 bit	0.95	0.59	0.65	0.30	0.57	0.22
252 bit × 32	30.2	18.6	20.8	9.5	18.9	6.8
252 bit × 128	121	74.3	83.5	37.8	76.1	27.4
252 bit × 512	484	297	358	165	302	109

Table 4.3.: Comparison of non-optimised prover runtime in seconds of aggregate range proofs from [BBB+18] with this work. Verification times are only included for completeness. See Section 4.5 for details.

Shuffle size	1000		10000		100000	
	P	V	P	V	P	V
Time [s]	8.8	4.4	117	56.1	1009	491

Table 4.4.: Evaluation of shuffle proofs via $\mathsf{QESA}_{\mathsf{Copy}}$ and $\mathsf{LMPA}_{\mathsf{simpleZK}}$.

almost identical.[23] This was also verified in independent benchmarks by Noether and Wedderburn.[24] We are not aware of similar optimisations for the prover.

Table 4.4 gives execution times for our shuffle proofs. They are an instantiation of [BG12], cf. Appendix B.2, and we project them to be 2–3× more computationally expensive than [BG12], but they are size $\mathcal{O}(\log(N))$ instead of $\mathcal{O}(\sqrt{N})$ for N ciphertexts. Again the very high execution times compared to [BG12] are caused by the underlying library.

[23] Application of delayed batch verification with multi-exponentiation to our setting is slightly different. However, compared to the costs of the multi-exponentiation, the difference is likely not noticeable.

[24] Code available at `https://github.com/crate-crypto/qesa/blob/master/src/ipa/no_zk.rs`

5. On Expected Polynomial Runtime in Cryptography

This chapter is taken verbatim from [Klo21] with minor changes.

5.1. Introduction

Interactive proof systems allow a prover P to convince a verifier V of the "truth" of a statement $\mathbb{x}$, i.e. that $\mathbb{x} \in \mathcal{L}$ for some language $\mathcal{L}$. Soundness of the protocol ensures that if the verifier accepts, then $\mathbb{x} \in \mathcal{L}$ with high probability. *Zero-knowledge* proof systems allow P to convince V of $\mathbb{x} \in \mathcal{L}$ *without revealing anything else.* The definition of zero-knowledge relies on the (more general) simulation paradigm: It stipulates that, for every (malicious) verifier V^*, there is a simulator Sim which, given only the inputs $\mathbb{x}$, *aux* of V^*, can produce a *simulated* output (or *view*[1]) *out* $= \mathsf{Sim}(\mathbb{x}, aux)$, which is indistinguishable from the output $\mathsf{out}_{\mathsf{V}^*}\langle \mathsf{P}(\mathbb{x}, \mathbb{w}), \mathsf{V}^*(\mathbb{x}, aux)\rangle$ of a real interaction. Thus, anything V^* learns in the interaction, it could simulate itself — *if* Sim *and* V^* *lie in the same complexity class.*

Let us write X/Y (zero-knowledge) for adversary complexity X and simulator complexity Y. The two widespread notions of zero-knowledge are PPT/PPT and PPT/EPT. The former satisfies the "promise of zero-knowledge", but comes at a price. Barak and Lindell [BL04] show that it is impossible to construct *constant* round proof systems with *black-box* simulation and negligible soundness error in the plain model. Since *constant round black-box* zero-knowledge is attractive for many reasons, the relaxation of PPT/EPT zero-knowledge is common. However, this asymmetry breaks the "promise of zero-knowledge". The adversary *cannot* execute Sim, hence it cannot simulate the interaction. More concretely, this setting does not compose well. If we incorporate an EPT simulator into a (previously PPT) adversary, the new adversary is EPT. This common approach — constructing simulators for more complex systems from simulators of building blocks — therefore fails due to the asymmetry.

To remedy the asymmetry, we need to handle EPT adversaries. There are several sensible definitions of EPT adversaries, but the arguably most natural choice are *designated* EPT adversaries. That is, adversaries which only need to be *EPT when interacting with the protocol they are designed to attack.* Feige [Fei90] first considered this setting, and demonstrates significant technical obstacles against achieving security in the presence of such attacks.

The problems of EPT (and designated adversaries) are not limited to zero-knowledge, and extend to the simulation paradigm, e.g. multi-party computation.

Preliminary Conventions. Throughout, λ denotes the security parameter. We generally consider objects which are families (of objects) parameterized by λ, but often leave the dependency implicit. We abbreviate *systems of (interactive) machines (or algorithms)* by *system.* A system is *closed*, if it only expects λ as input, and produces some output. For example, a prover P does not constitute a closed system, nor does the interaction $\langle \mathsf{P}, \mathsf{V}\rangle$, since it still lacks the inputs to P and V. Our primary setting

[1] We use view and output synonymously in the introduction.

is *uniform complexity* [Gol93], where inputs to an (otherwise closed) system are generated efficiently by so-called *input generators*. Interaction of algorithms A, B is denoted $\langle \mathsf{A}, \mathsf{B} \rangle$, the time spent in A is denoted $\mathsf{time}_{\mathsf{A}}(\langle \mathsf{A}, \mathsf{B} \rangle)$, and similarly for time spent in B or A + B. Oracle access to $\mathcal{O}$ is written $\mathsf{A}^{\mathcal{O}}$. An algorithm A is *a priori* efficient, if the runtime bound is independent from its environment, e.g. classical "a priori PPT". The term *a posteriori* emphasizes an absence of a priori efficiency, i.e. bounds which depend on the environment, e.g. in the case of designated adversaries.

5.1.1. Obstacles

We first recall some obstacles regarding expected runtime and designated adversaries which we have to keep in mind. For more discussions and details, we refer to the excellent introductions of [KL08; Gol10] and to [Fei90, Section 3].

Runtime Squaring. Consider (a family of) random variables T_λ over $\mathbb{N}$, where $\Pr[T_\lambda = 2^\lambda] = 2^{-\lambda}$ and T_λ is 0 otherwise. Then T_λ has polynomially bounded expectation $\mathbb{E}[T_\lambda] = 1$, but $\mathbb{E}[T_\lambda^2] = 2^\lambda$. That is $S_\lambda = T_\lambda^2$ is *not* expected polynomial time anymore. This behaviour not only prevents machine model independence of EPT as an efficiency notion, but also the non-black-box simulation technique of Barak [Bar01] (which suffers from a quadratic growth in runtime).

Composition and Rewinding. Consider an oracle algorithm $\mathsf{A}^{\mathcal{O}}$ with access to a PPT oracle $\mathcal{O}$. Then to check if the total time $\mathsf{time}_{\mathsf{A}+\mathcal{O}}(\mathsf{A}^{\mathcal{O}})$ is PPT, we can count an oracle call as a single step. Moreover, it makes no difference if A has "straightline" or "rewinding" access to $\mathcal{O}$. For EPT, even a standalone definition of "$\mathcal{O}$ is EPT" is non-trivial and possibly fragile. For example, there are oracles, where any PPT A with "straightline" access to $\mathcal{O}$ results in an EPT interaction, yet access "with rewinding" to $\mathcal{O}$ allows an explosion of expected runtime. See [KL08] for a concrete example.

Designated EPT Adversaries. For a **designated adversary** $\mathcal{A}$ against zero-knowledge of a proof system (P, V), we require (only) that $\mathcal{A}$ is *efficient when interacting with that protocol.* Since a zero-knowledge simulator *deviates* from the real protocol, the runtime guarantees of $\mathcal{A}$ are void.

5.1.2. Motivation: Reproving Zero-Knowledge of Graph 3-Colouring

The constant-round black-box zero-knowledge proof of Goldreich and Kahan [GK96] is our running example for demonstrating problems and developing our approach.

Recall that (non-interactive) commitment schemes allow a committer to commit to a value in a way which is *hiding* and *binding*, i.e. the commitment does not reveal the value to the receiver, yet it can be unveiled to at most one value. A commitment scheme consists of algorithms $(\mathsf{Setup}, \mathsf{Com}, \mathsf{VfyOpen})$. The *commitment key* is generated via $ck \xleftarrow{\$} \mathsf{Setup}(\lambda)$. For details, see Appendix C.2.

5.1.2.1. The Constant Round Protocol of Goldreich–Kahan

The protocol of [GK96] uses two different commitments, $\mathsf{Com}^{(H)}$ is perfectly hiding, $\mathsf{Com}^{(B)}$ is perfectly binding. The idea of protocol $\mathrm{G3C}_{\mathrm{GK}}$ is a parallel, N-fold, repetition of the standard zero-knowledge proof for G3C, with the twist that the verifier commits to all of its challenges beforehand. Let $G = (V, E)$ be the graph and let ψ be a 3-colouring of G. The prover is given (G, ψ) and the verifier G.

(P0) P sends $ck_{hide} \leftarrow \mathsf{Setup}^{(H)}(\lambda)$. ($ck_{bind} \leftarrow \mathsf{Setup}^{(B)}(\lambda)$ is deterministic.)

(V0) V picks $N = \lambda \cdot \mathsf{card}(E)$ challenge edges $e_i \leftarrow E$, and commits to them using $\mathsf{Com}^{(H)}$.

(P1) P picks randomized colourings for each of the N parallel repetitions of the standard graph 3-colouring proof system, and sends the $\mathsf{Com}^{(B)}$-committed randomized node colours to V.

(V1) V opens all commitments (to e_i).

(P2) P aborts if any opening is invalid. Otherwise, P proceeds in the parallel repetition using these challenges, i.e. in the i-th repetition P opens the committed colours for the nodes of edge e_i.

(V2) V aborts iff any opening is invalid, any edge not correctly coloured, or if ck_{hide} is "bad". Else V accepts.

The soundness of this protocol follows from $\mathsf{Com}^{(H)}$ being perfectly hiding. Therefore, each of the N parallel repetitions is essentially an independent repetition of the usual graph 3-colouring proof. For $N = \lambda \cdot \mathsf{card}(E)$ parallel rounds, the probability to successfully cheat is negligible (in λ), see [GK96].

5.1.2.2. Proving Zero-Knowledge: A (Failed?) Attempt

Now, we prove black-box zero-knowledge for *designated adversaries*. That is, we describe a simulator which uses the adversary V* only as a black-box, which can be queried and rewound to a (previous) state. We proceed in three game hops, gradually replacing the view of a real interaction with a simulated view. Successive games are constructed so that their change in output (which is a purported view) is indistinguishable.

G_0 This is the real G3C protocol. The output is the real view.

G_1 The prover rewinds a verifier which completes 5.1.2.1 successfully (i.e. sends *valid* openings on the first try) to 5.1.2.1 and repeats 5.1.2.1 until a second run where V validly opens all commitments. The output is the view of this second succesful run. The prover uses fresh randomness in each reiteration of 5.1.2.1 (whereas the black-box has fixed randomness).

G_2 If the two openings in 5.1.2.1 differ, return `ambig`, indicating ambiguity of the commitment. Otherwise, proceed unchanged.

G_3 The initial commitments (in 5.1.2.1) to a 3-colouring are replaced with commitments to 0. These commitments are never opened. In successive iterations, the commitments to a 3-colouring are replaced by commitments to pseudo-colourings ψ_i (for e_i), i.e. for edge $e_i = (u_i, v_i)$, ψ_i colours u_i and v_i differently (and uniformly), whereas ψ_i colours all $v \neq u_i, v_i$ with 0. Hence the opened commitments simulate a valid 3-colouring at the challenge edges e_i.

Evidently, Game G_3 outputs a purported view independent of the witness. Thus, the simulator is defined as in G_3: In a first try, it commits (using $\mathsf{Com}^{(B)}$) to all zeroes instead of a 3-colouring in 5.1.2.1, and uses this "garbage" commitment to learn the verifier's challenge (in 5.1.2.1). If the verifier does not successfully open the commitments (in 5.1.2.1), Sim aborts (as an honest prover would) and outputs the respective view. Otherwise, Sim rewinds the verifier to Step 2 and sends a pseudo-colouring (w.r.t. the previously revealed challenge) instead. Sim retries until the verifier succesfully unveils (in 5.1.2.1) again. (If the verifier opens to a different challenge, return $\mathit{view} = \mathtt{ambig}$.)

Now, we sketch a security proof for Sim. We argue by game hopping.

G_0 **to** G_1**.** The expected number of rewinds is at most 1. Namely, if V^* opens in 5.1.2.1 with probability ε, then an expected number of $\frac{1}{\varepsilon}$ rewinds are required. Consequently, the expected runtime is polynomial (and G_1 is EPT). The output distribution of the games is identical.

G_1 **to** G_2**.** It is easy to obtain an adversary against the binding property of $\mathsf{Com}^{(H)}$ which succeeds with the same probability that G_2 outputs $\mathtt{ambig}$. Thus, this probability is negligible.

G_2 **to** G_3**.** Embedding a (multi-)hiding game for $\mathsf{Com}^{(B)}$ in this step is straightforward. Namely, using the left-or-right indistinguishability formulation, where the commitment oracle either commits the first or second challenge message. Thus, by security of the commitment scheme, G_2 and G_3 are indistinguishable.[2]

A Closer Look. The above proof is clear and simple. But the described simulator is not EPT! While G_2 and G_3 are (computationally) indistinguishable, the transition *does not necessarily preserve expected polynomial runtime* [Fei90; KL08]. Feige [Fei90] points out a simple attack, where V^* brute-forces the commitments with some tiny probability p, and runs for a very long time if the contents are not valid 3-colourings. This is EPT in the real protocol, but our simulator as well as the simulator in [GK96] do not handle V^* in EPT. The problem lies with *designated* adversaries as following example shows.

Example 5.1.1. Let V^* sample in step 5.1.2.1 a "garbage" commitment c' to zeroes, using $\mathsf{Com}^{(B)}$ just like Sim in its first step, trying to predict Sim's choice. (c' is a "proof of simulation".) Now V^* unveils e in 5.1.2.1 if and only if it receives c'. The honest prover always aborts in 5.1.2.1 because V^* will never unveil. However, if Sim happens to chose $c = c'$ as its "garbage" commitment, the simulation runs forever, because V^* unveils only for this c', which is not a pseudo-colouring.

As described, V^* is a priori PPT, and indeed, the simulator in [GK96] uses a "normalization technique" which prevents this attack. However, exploiting *designated* PPT, V^* may instead run for a very long time, when it receives c'.

Obstructions to Simple Fixes. Let us recall a few simple, but insufficient fixes. A first idea is to *truncate* the execution of $\mathcal{A}$ at some point. For PPT adversaries, this may seem viable.[3] However, there are EPT adversaries, or more concretely runtime distributions, where *any strict polynomial truncation* affects the output in the real protocol *noticeably.* So we cannot expect that such a truncation works well for Sim. See [Fei90, Section 3] for a more convincing argument against truncation.

[2] We rely on security of binding and hiding against *expected time* adversaries, which follows from PPT-security by runtime truncation arguments, e.g. by Lemma 5.1.2.

[3] Even there, the situation is far from easy. In a UC setting with an *a posteriori* efficiency notion (and designated adversaries), Hofheinz, Unruh, and Müller-Quade show in [HUM13, Section 9] that (pathological) functionalities can make simulation in PPT impossible (if one wants security under composition for just a single instance).

Being unable to truncate, we could enforce better behaviour on the adversary. Intuitively, it seems enough to require that V^* runs in expected polynomial time *in any interaction* [KL08; Gol10]. However, even this is not enough. Katz and Lindell [KL08] exploit the soundness error of the proof system to construct an adversary which runs in expected polynomial time in any interaction, but still makes the expected runtime of the simulator superpolynomial. The problem is that these runtime guarantees are void in the presence of *rewinding*.

Modifications of these fixes work, but at a price: Katz and Lindell [KL08] use *superpolynomial* truncation and need to assume *superpolynomial* hardness. Goldreich [Gol10] *restricts* to algorithms (hence adversaries) which behave well under rewinding. We discuss these in Section 5.1.5. Our price will be proof techniques, which become more technical and, perhaps, more limited.

Our Fix: There is no Problem. Our starting point is *the conviction* that the given "proof" should *evidently* establish the security of the scheme for any *cryptographically sensible* notion of runtime. If one could *distinguish* the runtime of G_2 and G_3, then this would break the hiding property of the commitment scheme. Thus, the *runtimes are indistinguishable.* Following, in computational spirit, Leibniz' "identity of indiscernibles", we declare runtimes which are *indistinguishable from efficient* by efficient distinguishers as *efficient per definition.* With this, the proof works and the simulator, while not expected polynomial time, is *computationally expected polynomial time* (CEPT), which means its runtime distribution is indistinguishable from EPT.

We glossed over a crucial detail: We solved the problem with the very strategy we claim to fix — different runtime classes for Sim and V^*! Fortunately, Sim also handles CEPT adversaries in CEPT.

5.1.3. Contribution

Our main contribution is the reexamination of the notion of runtime in cryptography. We offer a novel, and arguably natural, alternative solution for a problem that was never fully resolved. Our contribution is therefore primarily of explorational and definitional nature. More concretely:

- We define CEPT, a small relaxation of EPT with a convenient characterization.
- To the best of our knowledge, this is the first work which embraces *uniform*[4] complexity, *expected* time, and *designated* adversaries.
- We develop general tools for this setting, most importantly, a hybrid lemma.
- Easy-to-check criteria show that many (all known?) black-box zero-knowledge arguments from standard assumptions in the plain model[5] have CEPT simulators which handle designated CEPT adversaries. Consequently, security against designated adversaries is natural. For example, the proof systems [GMW86; GK96; Lin13; Ros04; KP01; PTV14] satisfy our criteria.
- We impose no (non-essential) restrictions on the adversary, nor do we need additional (hardness) assumptions.
- We sketch the application of our techniques to secure function evaluation (SFE), and demonstrate that auxiliary input security implies modular sequential composition.

[4] Our results are applicable to a minor generalization of the non-uniform setting as well, namely non-uniformly generated input *distributions*, see Appendix C.5.1.2.

[5] Unfortunately, problems might arise with superpolynomial hardness assumptions, see Section 5.8.

All of this comes at a price. Our notions and proofs are not complicated, yet somewhat technical. This is, in part, because of a posteriori runtime and uniform complexity. Still, we argue that we have demonstrated the viability of our new notion of efficiency, at least for zero-knowledge.

A Complexity Theoretic Perspective. This work is only concerned with the complexity class of feasible *attacks*, and does not assume or impose complexity requirements on protocols. Due to designated adversaries, the complexity class of adversaries is (implicitly) defined per protocol, similar to [KL08]. We bootstrap feasibility from complexity classes for (standalone) sampling algorithms, i.e. algorithms with no inputs except λ. Hence a (designated) adversary is feasible if the *completed system* of protocol and adversary (including input generation) is CEPT (or more generally, in some complexity class of feasible sampling algorithms).

The complexity class of simulators is relative to the adversary, and thus depends both on the protocol and the ideal functionality. Namely, feasibility of a simulator Sim means that if an adversary $\mathcal{A}$ is feasible (w.r.t. the protocol), then "$\mathsf{Sim}(\mathcal{A})$" is feasible (w.r.t. the ideal functionality).

Comments on our Approach. The uniform complexity setting drives complexity, yet is necessary, since a notion of time that depends on non-uniformity is rather pathological. Losing the power of non-uniformity (and strictness of PPT) requires many small adjustments to definitions.[6] Moreover, annoying technical problems with efficiency arise inadvertently, depending on formalizations of games and models. As in prior work, we mostly ignore them, but do point them out and propose solutions. They are easily fixed by adding "laziness", "indirection", or "caching".

An important point raised by a reviewer of TCC'20 is the "danger of zero-knowledge being trivialized" by "expanding the class of attacks", and a case for "moving towards knowledge tightness" (with which we fully agree). Many variations of zero-knowledge, from weak distributional [DNRS03; CLP15] to precise [MP06; DG12], exist. We argue that our notion is very close to the "standard" notion with EPT simulation, but allows designated (C)EPT adversaries. Indeed, it seems to gravitate towards "knowledge tightness" [Gol10], as seen by runtime explosion examples due to expectation.

5.1.4. Technical Overview and Results

We give an overview of our techniques, definitions, and results. Recall that we only consider runtimes for closed systems (which receive only λ as input and produce some output). W.r.t. *uniform complexity* and *designated adversaries*, i.e. adversaries which only need to be efficient in the real protocol [Fei90], closed systems are the default situation anyway. A **runtime class** $\mathcal{T}$ is a set of runtime distributions. A **runtime (distribution)** is a family $(T_\lambda)_\lambda$ of distributions T_λ over $\mathbb{N}_0$. We use *runtime* and *runtime distribution* synonymously. Computational $\mathcal{T}$-time indistinguishability of oracles and distributions is defined in the obvious way (c.f. Section 5.2.6). For statistical $\mathcal{T}$-query indistinguishability, we count *only* queries as steps, and require $\mathcal{T}$-time w.r.t. this. (In our setting, unbounded queries often imply perfect indistinguishability, which is too strong.)

[6] For example, we need a stateful distinguisher for modular sequential security, whereas non-uniformly, state and even randomness can be trivially removed by coin-fixing, demonstrating the equivalence of many variations, whose equivalence in the uniform setting not clear. Thus, our definition of auxiliary input zero-knowledge deviates slightly from [Gol93].

5.1.4.1. The Basic Tools

Statistical vs. Computational Indistinguishability. The (folklore) *equivalence* of statistical and computational indistinguishability for distributions with *"small" support* is a simple, but central, tool. For polynomial runtime, "small" support means polynomial support, say $\{0, \dots, \mathsf{poly}_1(\lambda)\}$. Assuming non-uniform advice, the advice is large enough to encode the optimal decisions, achieving statistical distance as distinguishing advantage. This extends to "polynomially-tailed" runtime distributions T. There, by assumption, for any poly_0 there is a poly_1 such that $\Pr[T_\lambda > \mathsf{poly}_1(\lambda)] \leq \frac{1}{\mathsf{poly}_0(\lambda)}$, Hence, we can reduce to strict polynomial support by truncating at poly_1, sacrificing $1/\mathsf{poly}_0$ in statistical distance. The Markov bound shows that expected polynomial time is polynomially tailed. Removing non-uniformity is possible with repeated sampling, e.g. by approximating the distribution.

Standard Reduction. Another simple, yet central, tool is the *standard cutoff argument* (Section 5.4.1). It is the core tool to obtain *efficiency from indistinguishability.*

Lemma 5.1.2 (Standard reduction to PPT)**.** *Let $\mathcal{D}$ be a distinguisher for two oracles $\mathcal{O}_0$, $\mathcal{O}_1$. Suppose $\mathcal{D}$ has advantage at least $\varepsilon \geq \frac{1}{\mathsf{poly}_{\mathrm{adv}}}$ (infinitely often). Suppose furthermore that $\mathcal{D}^{\mathcal{O}_0}$ is EPT (even CEPT) with expected time* poly_0. *Then there is an a priori PPT distinguisher $\mathcal{A}$ with advantage at least $\frac{\varepsilon}{4}$ (infinitely often). (Here, ε,* $\mathsf{poly}_{\mathrm{adv}}$, poly_0 *are functions in λ.)*

We stress that we require *no runtime guarantees* for $\mathcal{D}^{\mathcal{O}_1}$ — it may never halt for all we know. For a proof sketch, define $N = 4\mathsf{poly}_0 \cdot \mathsf{poly}_{\mathrm{adv}}$ and let $\mathcal{A}$ be the runtime cutoff of $\mathcal{D}$ at N. The outputs of $\mathcal{A}^{\mathcal{O}_0}$ and $\mathcal{D}^{\mathcal{O}_0}$ are $\frac{\varepsilon}{4}$ close. For $\mathcal{A}^{\mathcal{O}_1}$ and $\mathcal{D}^{\mathcal{O}_1}$ this may be false. However, if $\mathcal{D}^{\mathcal{O}_1}$ exceeds N steps with probability higher than $\frac{2\varepsilon}{4}$, then the runtime is a distinguishing statistic with advantage $\frac{\varepsilon}{4}$. Thus, we can assume the outputs of $\mathcal{A}^{\mathcal{O}_1}$ and $\mathcal{D}^{\mathcal{O}_1}$ are $\frac{2\varepsilon}{4}$ close. Now, a short calculation shows that $\mathcal{A}$ has advantage at least $\frac{\varepsilon}{4}$. Namely, $\Delta(\mathcal{A}^{\mathcal{O}_1}, \mathcal{A}^{\mathcal{O}_0}) \geq \Delta(\mathcal{D}^{\mathcal{O}_1}, \mathcal{D}^{\mathcal{O}_0}) - \Delta(\mathcal{A}^{\mathcal{O}_1}, \mathcal{D}^{\mathcal{O}_1}) - \Delta(\mathcal{D}^{\mathcal{O}_0}, \mathcal{A}^{\mathcal{O}_0})$.

5.1.4.2. Computationally Expected Polynomial Time

We define the runtime classes $\mathcal{PPT}$ (resp. $\mathcal{EPT}$), as usual, i.e. $(T_\lambda)_\lambda \in \mathcal{PPT} \iff \exists \mathsf{poly}\colon \Pr[T_\lambda \leq \mathsf{poly}(\lambda)] = 1$ (resp. $(T_\lambda)_\lambda \in \mathcal{EPT} \iff \exists \mathsf{poly}\colon \mathbb{E}[T_\lambda] \leq \mathsf{poly}(\lambda)$).

Definition 5.1.3 (Simplified[7] Definition 5.3.5). A runtime S, i.e. a family of random variables S_λ with values in $\mathbb{N}_0$, is **computationally expected polynomial time (CEPT)**, if there exists a runtime T which is (perfectly) expected polynomial time (i.e. EPT), such that any a priori PPT distinguisher has negligible distinguishing advantage for the distributions T and S. The class of CEPT runtime distributions is denoted $\mathcal{CEPT}$. **Computationally strict polynomial time (CPPT)** is defined analogously.

Characterizing CEPT. At a first glimpse, CEPT looks hard to handle. Fortunately, this is a mirage. We have following characterization of CEPT.

Proposition 5.1.4 (Simplified[7] Corollary 5.3.9)**.** *Let T be a runtime. The following are equivalent:*

0. *T is in $\mathcal{CEPT}$.*

[7] Formally, "triple-oracle" instead of "standard" indistinguishability is used. Assuming non-uniform advice, or runtimes T, S which are induced by algorithms, the simplified definition is equivalent to the actual one.

1. *$\exists S \in \mathcal{EPT}$ which is* computationally *PPT-indistinguishable from T.*
2. *$\exists S \in \mathcal{EPT}$ s.t. T and S are statistically indistinguishable (given polynomially many samples).*
3. *There is a set of good events $\mathcal{G}_\lambda$ with $\Pr[\mathcal{G}_\lambda] \geq 1-\varepsilon(\lambda)$ such that $\mathbb{E}[T_\lambda \mid \mathcal{G}_\lambda] \leq t_\lambda$ (for the conditional expectation), where ε is negligible and t is polynomial.*

Let T be a runtime. Item 3 defines **virtually expected time** (t, ε) with *virtual expectation* (bounded by) t and *virtuality* ε. Thus, the characterization says that computational, statistical and virtual EPT coincide.

Proposition 5.1.4 follows essentially from the statistical-to-computational reduction and a variant of Lemma 5.1.2. Thanks to this characterization, working with CEPT is feasible. One uses item 1 to justify that indistinguishability transitions preserve CEPT. And one relies on item 3 to simplify to the case of EPT, usually in unconditional transitions, such as efficiency of rewinding.

An Intrinsic Characterization. The full Corollary 5.3.9 not only reveals that CEPT is "well-behaved". It also shows that the runtime class $\mathcal{CEPT}$ is "closed under indistinguishability": Any runtime S which is CEPT-indistinguishable from some $T \in \mathcal{CEPT}$ lies in $\mathcal{CEPT}$. This intrinsic property sets it apart from EPT. (Indeed, $\mathcal{CEPT}$ is the closure of $\mathcal{EPT}$.) PPT and CPPT behave analogously.

Example 5.1.5. Let A be an algorithm which outputs 42 in exactly 10^{10} steps, and let A′ act identical to A, except with probability $2^{-\lambda}$, in which case it runs $2^{2\lambda}$ steps. Then A′ is neither PPT nor EPT. Yet, A and A′ are indistinguishable even given *timed* black-box access. That is, observing both output and runtime of the black-box, it is not possible to tell A and A′ apart. Thus, it is rather unexpected that A′ is considered inefficient. For many properties, e.g. correctness or soundness, statistical relaxations from "perfect" exist. CPPT and CEPT should be viewed as such relaxations for efficiency.

Working with CEPT. Applying the characterization of CEPT to a whole system $\langle \mathsf{P}, \mathsf{V}^* \rangle$, the good event $\mathcal{G}$ may induce arbitrary stochastic dependencies on (internal) random coins of the parties. This is inconvenient. We are interested only in one party, namely V^*. Moreover, in a simulation, there is no P anymore and the probability space changed, hence there is no event $\mathcal{G}$. To account for this, we observe that only the messages V^* receives from P are relevant for V^*'s behaviour, not P's internal randomness. We formulate a convenience lemma (Lemma 5.3.12) for handling this. Roughly Lemma 5.3.12 states that for interacting algorithms $\langle \mathsf{A}, \mathsf{B} \rangle$, there is a modification B′ (which need not be efficiently computable), which immediately aborts "bad executions" by sending `timeout`. If the closed system $\langle \mathsf{A}, \mathsf{B} \rangle$ is CEPT, i.e. $\mathsf{time}_{\mathsf{A+B}}(\langle \mathsf{A}, \mathsf{B} \rangle)$ is CEPT, the probability for `timeout` is negligible. Then, by construction, $\mathsf{time}_{\mathsf{B'}}(\langle \mathsf{A}, \mathsf{B'} \rangle)$ will be EPT. This makes B′ into a convenient tool to track the evolution of runtime and virtuality under actions such as rewinding. By using B′ only via oracle-access, its possible inefficiency poses no problems. After the (runtime) analysis, oracle-access to B′ is replaced with B again. Importantly, B′ is just a means to reason about changes in runtime when applying rewinding to B. One can also reason without introducing B′, by using the analysis in Lemma 5.3.12 directly.

5.1.4.3. Definitions and Tools for Zero-knowledge

Here, we state our definition of uniform complexity zero-knowledge, demonstrate how to prove zero-knowledge for $\mathsf{G3C}_{\mathsf{GK}}$, and then abstract the approach to cover a large class of protocols.

Definition of Zero-Knowledge. For uniform auxiliary input zero-knowledge, the input $(\mathbb{x}, \mathbb{w}, \mathit{aux}, \mathit{state}) \leftarrow \mathcal{I}(\lambda)$ is efficiently generated by an *input generator* $\mathcal{I}$. A designated adversary $(\mathcal{I}, \mathsf{V}^*)$ consists of input generation, malicious verifier, and distinguisher, but we leave $\mathcal{I}$ often implicit. The distinguisher receives *out* and *state*, the latter is needed for modular sequential composition.[8] Here, $\mathit{out} = \mathsf{out}_{\mathsf{V}^*}\langle \mathsf{P}(\mathbb{x}, \mathbb{w}), \mathsf{V}^*(\mathbb{x}, \mathit{aux})\rangle$ or $\mathit{out} = \mathsf{out}_{\mathsf{Sim}}\mathsf{Sim}(\mathsf{code}(\mathsf{V}^*), \mathbb{x}, \mathit{aux})$, where $(\mathbb{x}, \mathbb{w}, \mathit{aux}, \mathit{state})$ is sampled by $\mathcal{I}(\lambda)$. As a shorthand, for the system which lets $\mathcal{I}$ sample inputs and passes them as above, we write $\langle \mathsf{P}, \mathsf{V}\rangle_{\mathcal{I}}$. From designated CEPT adversaries, we require that $\mathsf{time}_{\mathcal{I}+\mathsf{P}+\mathsf{V}^*+\mathcal{D}}((\mathit{state}, \mathsf{out}_{\mathsf{V}^*}\mathsf{P}(\mathbb{x}, \mathbb{w}), \mathsf{V}^*(\mathbb{x}, \mathit{aux})))$ is CEPT.

Concrete example. Recall that in Section 5.1.2, we showed zero-knowledge of the graph 3-colouring protocol $\mathrm{G3C_{GK}}$ of Goldreich and Kahan [GK96] as follows:

Step 1: Introduce all rewinding steps as in G_1. Here, virtually expected runtime and virtuality at most doubles. To see this, one can use Lemma 5.3.12 to "replace" V^* with an modified V' which yields an EPT execution and outputs `timeout` for "bad" queries. Since Game G_1 at most doubles the probability that some query *query* is asked, bad queries are only twice as likely, i.e. virtuality at most doubles. It is easy to see that the virtually expected runtime also (at most) doubles.

Step 2: Apply indistinguishability transitions, which reduce to hiding resp. binding properties of the commitment. From this, we obtain both good output quality and efficiency of Sim. Concretely, indistinguishability and efficiency follow by an application of the standard reduction (to PPT).

We abstract this strategy to cover a large class of zero-knowledge proofs.[9] Intuitively, we apply the ideas of [Gol10] ("normality") and [KL08] ("query indistinguishability"), but separate the unconditional part (namely, that rewinding preserves efficiency), and the computational part (namely, that simulated queries preserve efficiency).[10]

Abstracting Step 1 (Rewinding Strategies). A **rewinding strategy** RWS has black-box rewinding (bb-rw) access to a malicious verifier V^*, and abstracts a simulator's rewinding behaviour. Unlike the simulator, RWS has access to the witness. For RWS to be **normal**, we impose three requirements.

Firstly, a normal rewinding strategy outputs an adversarial view which is *distributed (almost) as in the real execution.* Secondly, there is some poly so that

$$\mathbb{E}[\mathsf{time}_{\mathsf{RWS}+\mathsf{V}^*}(\mathsf{RWS}^{\mathsf{V}^*})] \le \mathsf{poly}(\lambda) \cdot \mathbb{E}[\mathsf{time}_{\mathsf{P}+\mathsf{V}^*}(\langle \mathsf{P}, \mathsf{V}^*\rangle)]$$

for any adversary V^*. We call this (polynomial) **runtime tightness** of RWS.[11] Thirdly, RWS has (polynomial) **probability tightness**, which is defined as follows: Let $\mathrm{pr}_{\mathrm{rws}}(\mathit{query})$ be the probability that RWS asks V^* a query *query*. Let $\mathrm{pr}_{\mathrm{real}}(\mathit{query})$ be the probability that the prover P asks *query*. Then RWS has probability tightness poly if for all queries *query*

$$\mathrm{pr}_{\mathrm{rws}}(\mathit{query}) \le \mathsf{poly}(\lambda) \cdot \mathrm{pr}_{\mathrm{real}}(\mathit{query}).$$

Intuitively, runtime tightness ensures that RWS preserves EPT, whereas probability tightness bounds the growth of virtuality. Indeed, the virtuality δ in $\langle \mathsf{P}, \mathsf{V}^*\rangle$ increases to at most $\mathsf{poly} \cdot \delta$ in $\mathsf{RWS}^{\mathsf{V}^*}$. This

[8] While [Gol93] passes no extra *state*, only sequential *repetition* is proven there.

[9] Strictly speaking, we concentrate on zero-knowledge *arguments*, since we need efficient provers.

[10] We significantly deviate from [KL08] to obtain simpler reductions. See Appendix C.6 for an approach similar to [KL08].

[11] Up to minor technical details, polynomial runtime tightness of RWS coincides with "normality" of Sim in [Gol10, Def. 6].

follows because the probability for a "bad" query (a `timeout` of the modified V' from Lemma 5.3.12) in $\mathsf{RWS}^{\mathsf{V}^*}$ is at most poly-fold higher than in $\langle \mathsf{P}, \mathsf{V}^* \rangle$.

Lemma 5.1.6 (InformalLemma 5.6.5). *Let* RWS *be a normal rewinding strategy for* (P, V) *with runtime and probability tightness* poly. *Let* $(\mathcal{I}, \mathsf{V}^*)$ *be an adversary. If* $\langle \mathsf{P}, \mathsf{V}^* \rangle_{\mathcal{I}}$ *is CEPT with virtually expected time* (t, ε)*, then* $\mathsf{RWS}(\mathsf{V}^*)$ *composed with* $\mathcal{I}$ *is CEPT with virtually expected time* $(\mathrm{poly} \cdot t, \mathrm{poly} \cdot \varepsilon)$.

(Weak) Relative Efficiency. We generalize the guarantees of rewinding strategies to *relative efficiency* of (oracle) algorithms. An oracle algorithm B is **efficient relative to** A with **runtime tightness** $(\mathrm{poly}_{\mathsf{time}}, \mathrm{poly}_{\mathsf{virt}})$ if for all oracles $\mathcal{O}$: If $\mathsf{time}_{\mathsf{A}+\mathcal{O}}(\mathsf{A}^{\mathcal{O}})$ is virtually expected (t, ε)-time, then $\mathsf{time}_{\mathsf{B}+\mathcal{O}}(\mathsf{B}^{\mathcal{O}})$ is virtually expected $(\mathrm{poly}_{\mathsf{time}} \cdot t, \mathrm{poly}_{\mathsf{virt}} \cdot \varepsilon)$-time.

We call B **weakly efficient relative** to A, if whenever $\mathsf{time}_{\mathsf{A}+\mathcal{O}}(\mathsf{A}^{\mathcal{O}})$ is efficient (e.g. CEPT), then $\mathsf{time}_{\mathsf{B}+\mathcal{O}}(\mathsf{B}^{\mathcal{O}})$ is efficient (e.g. CEPT).

Abstracting Step 2 (Simple Assumptions). A **"simple" assumption** is a pair of efficiently computable oracles $\mathcal{C}_0$ and $\mathcal{C}_1$, and the assumption that $\mathcal{C}_0 \overset{c}{\approx} \mathcal{C}_1$, i.e. $\mathcal{C}_0$ and $\mathcal{C}_1$ cannot be distinguished in PPT.[12] For example, hiding resp. binding for commitment schemes are simple assumptions.

In Step 2, we reduce the indistinguishability of $\mathsf{RWS}^{\mathsf{V}^*}$ and $\mathsf{Sim}^{\mathsf{V}^*}$ to a simple assumption. That is, there is some algorithm R such that $\mathsf{RWS}^{\mathsf{V}^*} \equiv \mathsf{R}^{\mathcal{C}_0}(\mathsf{V}^*)$, and $\mathsf{R}^{\mathcal{C}_1}(\mathsf{V}^*) \equiv \mathsf{Sim}^{\mathsf{V}^*}$. Moreover, we assume that $\mathsf{R}^{\mathcal{C}_0}(\mathsf{V}^*)$ is efficient relative to $\mathsf{RWS}^{\mathsf{V}^*}$, and $\mathsf{Sim}^{\mathsf{V}^*}$ is efficient relative to $\mathsf{R}^{\mathcal{C}_1}(\mathsf{V}^*)$.

Putting It Together (Benign Simulators). Black-box simulators whose security proof follows the above outline are called **benign**. See Fig. 5.1 for an overview of properties and their relation.

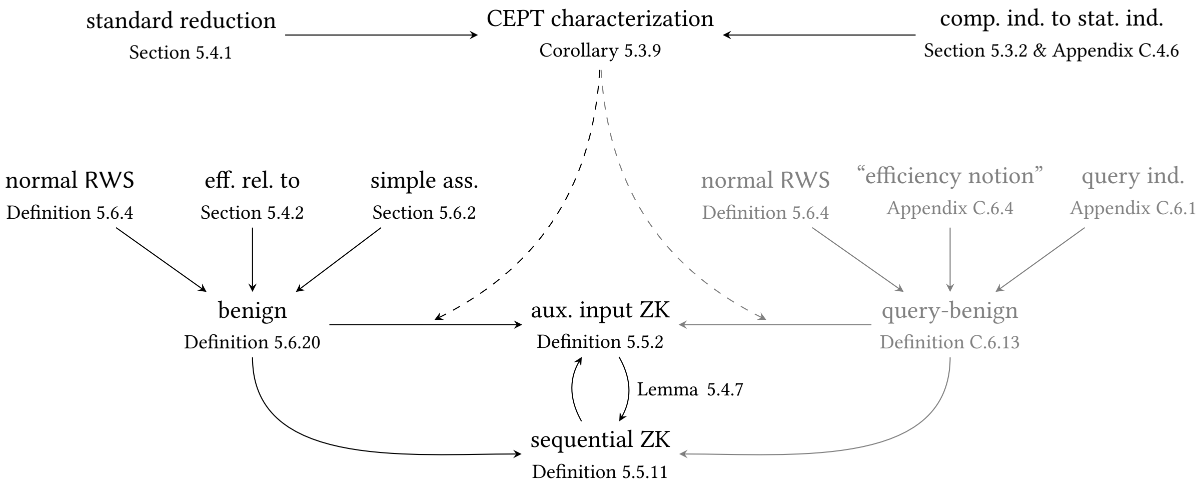


Figure 5.1.: A rough overview of dependencies of core results and definitions. The greyed out approach follows [KL08] more closely. The top line is used everywhere implicitly.

[12] Technically, our definition of "simple assumption" corresponds to falsifiable assumptions [Nao03] in the sense of [GW11]. We deliberately do not call them falsifiable, since our proof techniques should extend to a larger class of assumptions, which includes non-falsifiable assumptions.

Lemma 5.1.7 (Informal Lemma 5.6.23)**.** *Argument systems with* benign *simulators are* auxiliary-input zero-knowledge *against CEPT adversaries.*

Proof summary. The proof strategy above can be summarized symbolically:

$$\mathsf{out}_{\mathsf{V}^*}\langle \mathsf{P}, \mathsf{V}^*\rangle \equiv \mathsf{RWS}(\mathsf{V}^*) \equiv \mathsf{R}^{\mathcal{C}_0}(\mathsf{V}^*) \overset{c}{\approx} \mathsf{R}^{\mathcal{C}_1}(\mathsf{V}^*) \equiv \mathsf{Sim}(\mathsf{V}^*).$$

More precisely, consider a CEPT adversary $(\mathcal{I}, \mathsf{V}^*)$. By normality of RWS, $\mathsf{out}_{\mathsf{V}^*}\langle \mathsf{P}, \mathsf{V}^*\rangle$ and $\mathsf{RWS}(\mathsf{V}^*)$ have (almost) identical output distributions, and $\mathsf{RWS}(\mathsf{V}^*)$ is CEPT. By relative efficiency, $\mathsf{R}^{\mathcal{C}_0}(\mathsf{V}^*)$ is CEPT if $\mathsf{RWS}^{\mathsf{V}^*}$ is CEPT. Since $\mathcal{C}_0 \overset{c}{\approx} \mathcal{C}_1$, by a standard reduction, if $\mathsf{R}^{\mathcal{C}_0}(\mathsf{V}^*)$ is CEPT, so is $\mathsf{R}^{\mathcal{C}_1}(\mathsf{V}^*)$, and their outputs are indistinguishable. Finally, since $\mathsf{Sim}^{\mathsf{V}^*}$ is efficient relative to $\mathsf{R}^{\mathcal{C}_1}(\mathsf{V}^*)$, also $\mathsf{Sim}^{\mathsf{V}^*}$ is CEPT. All in all, $\mathsf{Sim}^{\mathsf{V}^*}$ is efficient and produces indistinguishable outputs. □

Benign simulators are common, e.g. the classic, constant round, and concurrent zero-knowledge protocols in [GMW86; GK96; Lin13; Ros04; KP01; PTV14] satisfy this property.

5.1.4.4. Sequential Composition and Hybrid Arguments

It turns out that hybrid arguments are non-trivial in the setting of a posteriori efficiency. Here, we outline the challenges in proving the hybrid lemma, how to overcome them, and how to obtain security of sequential composition from our abstract hybrid lemma.

Intermezzo: Tightness Bounds. The use of relative efficiency with polynomial tightnesss bounds is not strictly necessary. Nevertheless, it offers "more quantifiable" security and is easier to handle. For example, benign simulators are easily seen to "compose sequentially" because, (1) normal RWS and relative efficiency compose sequentially, and (2) "simple" assumptions satisfy indistinguishability under "repeated trials". Together, this translates to sequential composition of benign simulation. Hence, argument systems with *benign* simulators are *sequential zero-knowledge* against CEPT adversaries. Unfortunately, the general case is much more involved.

The Hybrid Lemma. To keep things tidy, we consider an abstract hybrid argument, which applies to zero-knowledge simulation and much more. Due to a posteriori efficiency, the lemma is both non-trivial to prove and non-trivial to state.

Lemma 5.1.8 (Lemma 5.4.7)**.** *Let $\mathcal{O}_0$ and $\mathcal{O}_1$ be two oracles and suppose that $\mathcal{O}_1$ is weakly efficient relative to $\mathcal{O}_0$ and $\mathcal{O}_0 \overset{c}{\approx} \mathcal{O}_1$. Denote by* $\mathsf{rep}(\mathcal{O}_0)$ *and* $\mathsf{rep}(\mathcal{O}_1)$ *oracles which give repeated access to independent instances of $\mathcal{O}_b$. Then* $\mathsf{rep}(\mathcal{O}_1)$ *is weakly efficient relative to* $\mathsf{rep}(\mathcal{O}_0)$ *and* $\mathsf{rep}(\mathcal{O}_0) \overset{c}{\approx} \mathsf{rep}(\mathcal{O}_1)$.

Lemma 5.1.8 hides much of the complexity caused by a posteriori efficiency, and is often a suitable black-box drop-in for the hybrid argument. We sketch how to adapt the usual hybrid reduction. In our setting, $\mathsf{rep}(\mathcal{O}_b)$ gives access to arbitrarily many independent instances of $\mathcal{O}_b$. The usual hybrids H_i use $\mathcal{O}_1$ for the first i instances, and switch to $\mathcal{O}_0$ for all other instances. W.l.o.g., only $q = \mathsf{poly}(\lambda)$ many $\mathcal{O}$-instances are accessed by the distinguisher $\mathcal{D}$. The hybrid distinguisher $\mathcal{D}'$ guesses an index $i^* \leftarrow \{0, \ldots, q-1\}$, and simulates a hybrid H_{i+b} embedding its challenge oracle $\mathcal{O}_b^*$.

If $\mathcal{D}$ has advantage ε, then the hybrid distinguisher $\mathcal{D}'$ has advantage ε/q. In the classic PPT setting, we assume that $\mathcal{O}_0$ and $\mathcal{O}_1$ are classical PPT, and hence find that $\mathcal{D}'$ is PPT and therefore efficient. In an a posteriori setting, the efficiency of $\mathcal{D}'$ is a bigger hurdle. We make the minimal assumptions, that $\mathsf{time}_{\mathcal{D}+\mathsf{rep}(\mathcal{O}_0)}(\mathcal{D}^{\mathsf{rep}(\mathcal{O}_0)})$ is efficient and that $\mathcal{O}_1$ is weakly efficient relative to $\mathcal{O}_0$.[13] Hence, we do not trivially know whether $\mathsf{time}_{\mathcal{D}+\mathsf{rep}(\mathcal{O}_1)}(\mathcal{D}^{\mathsf{rep}(\mathcal{O}_1)})$ or the hybrid distinguisher $\mathcal{D}'$, which has to emulate many oracle instances, is efficient. Indeed, a naive argument would invoke weak relative efficiency q times. In the case of PPT, this would mean q-many polynomial bounds. But, for all we know, these could have the form $2^i \mathsf{poly}(\lambda)$ in the i-th invocation, leading to an inefficient simulation.

The core problem is therefore to avoid a superconstant application of weak relative efficiency.[14] Essentially this problem was encountered by Hofheinz, Unruh, and Müller-Quade [HUM13] in the setting of universal composability and a posteriori PPT. They provide a nifty solution, namely to *randomize the oracle indexing*. This ensures that, in each hybrid, every emulation of $\mathcal{O}_0$ (resp. $\mathcal{O}_1$) has identical runtime distribution T_0 (resp. T_1). This gives a uniform bound on runtime changes. Now, we show how to extend the proof of [HUM13], which is limited to CPPT.

We prove the hybrid argument in game hops, starting from the real protocol G_1. In G_2, we replace *one* oracle instance of $\mathcal{O}_0$ by $\mathcal{O}_1$ (at a random point). In G_3, every instance of $\mathcal{O}_0$ but one is replaced by $\mathcal{O}_1$. In G_4, only $\mathcal{O}_1$ is used. Since $\mathcal{O}_1$ is weakly efficient relative to $\mathcal{O}_0$ and $\mathcal{O}_0 \overset{c}{\approx} \mathcal{O}_1$, the transitions from G_1 to G_2 (resp. G_3 to G_4) preserve efficiency and are indistinguishable. The step from G_2 to G_3 is the crux. Note that we have at least one $\mathcal{O}_0$ (resp. $\mathcal{O}_1$) instance in either game. Take any one and denote the time spent in that instance by T_0 (resp. T_1). Since we randomized the instances, the distribution of T_0 (resp. T_1) does not depend on the concrete instance. Importantly, even in the hybrid *reduction*, there is an instance which can be used to compute T_0 (resp. T_1). Moreover, the total time spent in computing instances of $\mathcal{O}_0$ and $\mathcal{O}_1$ is "dominated"[15] by $q \cdot T_0 + q \cdot T_1$. Thus, it suffices to prove that $S = T' + T_0 + T_1$ is CEPT, where T' is the time spent outside emulation of instances of $\mathcal{O}_0$ and $\mathcal{O}_1$. (Note that S, T', T_0, T_1 depend on the hybrid H_ℓ, where $\ell \in \{1, \dots, q-1\}$; we suppressed this dependency.) Now, we have two properties:

- S_ℓ is CEPT if and only if $\mathsf{time}(\mathsf{H}_\ell)$ is CEPT for the ℓ-th hybrid H_ℓ.
- The reduction can compute and output S_ℓ.

Thus, it suffices that S_1 and S_{q-1} are indistinguishable, since we know that S_1 is CEPT. Curiously, we now reduced efficiency to indistinguishability.[16] To prove indistinguishability, we can truncate the reduction (or rather, the hybrids) to strict PPT as in the standard reduction. Thus, we obtain $S_1 \overset{c}{\approx} S_q$. The hybrid lemma follows. The actual reasoning of this last step is a bit lengthier, but follows [HUM13] quite closely: We truncate each oracle separately to maintain symmetry of `timeout` probabilities. Unfortunately, the reduction does not give the usual telescoping sum, since the challenge oracle cannot be truncated. Due to symmetry, the error is "dominated" by observed `timeout`s. Hence, it suffices to find a (uniform) bound for the `timeout` probabilities over all H_ℓ. Our reasoning for this is mildly more complex than [HUM13], since we do not have negligible bounds for `timeout`s, but only polynomial tail bounds, and we make a weaker assumption on efficiency of $\mathcal{O}_0$ and $\mathcal{O}_1$.

Modular Sequential Composition. With Lemma 5.1.8 at hand, it is straightforward to prove that auxiliary input zero-knowledge composes sequentially. In fact, the well-known proof works almost without

[13] The hybrid proof technique requires the hybrid distinguisher to emulate all but one oracle instance, and for this we need weak relative efficiency.

[14] For reference, even for a priori PPT sequential composition for zero-knowledge, one must avoid a superconstant invocation of the existence of simulators. There, the solution is to consider a "universal" adversary and its "universal" simulator.

[15] To be exact, dominated with slack q: $\Pr[\mathsf{time}_{\mathcal{O}_0+\mathcal{O}_1}(\mathsf{H}_\ell) > t] \leq q \cdot \Pr[q(T_{\ell,0} + T_{\ell,1}) > t]$.

[16] The CEPT characterization (Corollary 5.3.10) does not strictly apply here, but a simple variation does.

modifications by using the hybrid lemma (Lemma 5.1.8), which absorbs the bulk of the complexity. Indeed, it is possible to prove a modular sequential composition theorem for secure function evaluation, similar to [KL08]. Interestingly, in [KL08], subprotocols must have simulators which are EPT *in any interaction*, whereas in our setting, there is no such restriction.

5.1.5. Related Work

We are aware of three (lines of) related works w.r.t. EPT: The results by Katz and Lindell [KL08] and those of Goldreich [Gol10], both focused on cryptography. And the relaxation of EPT for average-case complexity by Levin [Lev86]. A general difference of our approach is, that we treat the security parameter separate from input sizes, whereas [KL08; Gol10] assume $\lambda = |\mathbb{x}|$.[17] With respect to a posteriori runtime, [HUM13] is a close analogue, although for PPT and in the UC setting.

Comparison with [KL08]. Katz and Lindell [KL08] tackle the problem of expected polynomial time by using a *superpolynomial runtime cutoff.* They show that this cutoff guarantees a (strict) EPT adversary. However, for the superpolynomial cutoff, they need to *fix* one superpolynomial function α and have to assume security of primitives w.r.t. (strict) α-time adversaries. Squinting hard enough, their approach is dual to ours. Instead of assuming superpolynomial security and doing a cutoff, we "ignore negligible events" in runtime statistics, thus doing a "cutoff in the probability space". Moreover, we require no fixed bound.

Interestingly, their first result [KL08, Theorem 5] holds for "adversaries which are EPT w.r.t. the real protocol". Their notion is minimally weaker than ours, as it requires efficiency of the adversary *for all inputs* instead of a sequence of input distributions.[18] [KL08, Section 3.5] claims that other scenarios, e.g. sequential composition, fall within [KL08, Theorem 5]. Their *modular* sequential composition theorem, [KL08, Theorem 12], however, requires that subprotocol simulators are "expected polynomial time *in any interaction*", which neither Theorem 5 nor Theorem 12 assert for the resulting simulators.

Comparison with [Gol10]. Goldreich [Gol10] strengthens the notion of expected polynomial time to obtain a complexity class which is stand-alone and suitable for rewinding based proofs. He requires *expected polynomial time w.r.t. any reset attack*, hence restricts to "nice" adversaries. With this, normal (in the sense of [Gol10]) black-box simulators run in expected polynomial time, essentially by assumption. This way of dealing with designated adversaries is far from the spirit of our work.

Comparison with [Lev86]. The relaxation of expected polynomial time adopted by Levin [Lev86] and variations [Gol11b; Gol10; BT06] are very strong. Let T be a runtime distribution. One definition requires that for some poly and $\gamma > 0$, $\Pr[T_\lambda > C] \le \frac{\mathsf{poly}(\lambda)}{C^\gamma}$ for large enough λ and $C \ge 0$. Equivalently, $\mathbb{E}[T_\lambda^\gamma]$ is polynomially bounded (in λ) for some $\gamma > 0$. Allowing negligible "errors" relaxes the notion further. This definition fixes the composition problems of expected polynomial time. But arguably, it stretches what is considered efficient far beyond what one may be willing to accept. Indeed, runtimes whose expectation is "very infinite" are considered efficient.[19] The goals of average case complexity

[17] For completeness, we show how to mirror this weakened security in Appendix C.5.4.3.

[18] Their definitions are a consequence of their non-uniform security definition and complexity setting. The proof of [KL08, Theorem 5] never changes adversarial inputs, so there is no obstruction to handling designated adversaries in our sense.

[19] Setting $c = 2$ and $\gamma = 3$ in Remark 5.1.9 yields a runtime T with $\mathbb{E}[T] = \sum_{n=1}^{\infty} n$, which is still considered efficient. (The limit $-\frac{1}{12}$ is not applicable here.)

theory and cryptography do not align here. We stress that our approach, while relaxing expected polynomial time, is far from being so generous, see Section 5.1.6.1. (For completeness, we note that we are not aware of work on designated adversaries in this setting.)

Related Work on CPPT. The notion of CPPT is (in different forms) used and well-known. For example, Boneh and Shoup [BS20] rely on such a notion. This sidesteps technical problems, such as sampling uniformly from $\{0, 1, 2\}$ with binary coins. With a focus on complexity theory, Goldreich [Gol11a] defines *typical efficiency* similar to CPPT. As the relaxations for strict bounds is very straightforward, we suspect more works using CPPT variations for a variety of reasons.

Comparison with [HUM13]. Hofheinz, Unruh, and Müller-Quade [HUM13] define *PPT with overwhelming probability (w.o.p.)*, i.e. CPPT, and consider a posteriori efficiency. They work in the setting of universal composability (UC), and their main focus is an overall sensible notion of runtime, which does not artificially restrict evidently efficient *functionalities*, such as databases or bulletin boards. Their notion of efficiency is similar to our setting with CPPT. In fact, we use their techniques for the hybrid argument. Since [HUM13] defines and assumes *protocol efficiency*, which we deliberately neglect, there are some differences. Reinterpreting [HUM13], their approach is based on: "If *for all* (stand-alone) efficient $\mathcal{D}$ the machine $\mathcal{D}^{\mathcal{O}_0}$ is efficient, then *for all* (stand-alone) efficient $\mathcal{D}$ the machine $\mathcal{D}^{\mathcal{O}_1}$ is efficient."[20] Our approach is based on: "*For all* $\mathcal{D}$, if the machine $\mathcal{D}^{\mathcal{O}_0}$ is efficient, then the machine $\mathcal{D}^{\mathcal{O}_1}$ is efficient." The stronger (protocol) efficiency requirements are harder to justify in our setting. (Even classical PPT $\mathcal{O}_0$ can be "inefficient" for *expected* poly-size inputs. E.g., disallowing quadratic time protocols seems harsh.)

More Related Work. Halevi and Micali [HM98] define a notion of efficiency for extractors in proofs of knowledge, which closely resembles our notion of normal rewinding strategies. Precise zero-knowledge [MP06; Pas06] requires that simulation and real execution time are closely related. Due to Feige's "attack" (or Example 5.1.1), this does not seem to help with designated EPT adversaries.

5.1.6. Separations

We briefly provide separations between some runtime notions. Here, we focus only on efficiency of adversaries, and *ignore* requirements imposed on protocol efficiency, since we deliberately neglected those. We consider *basic runtime classes* (i.e. runtimes of sampling algorithms) and how they are *lifted to interactive algorithms.*

Both [KL08, Definition 1] and [HUM13, Definitions 1 and 2] use an "a posteriori" lifting. The former lifts EPT, the latter lifts CPPT; both allow designated adversaries and are similar to our setting. "A priori" liftings, such as [Gol10, Definitions 1–4] are far more restrictive (on adversaries), effectively disallowing designated adversaries.

Regarding the underlying runtime classes, the works [KL08; Gol10] deal with (perfect) EPT, negligible deviations are not allowed. The notion of PPT w.o.p. from [HUM13] and CPPT coincide. To separate PPT, EPT, CPPT, CEPT, and Levin's relaxations, we first recall fat-tailed distributions.

[20] Think of $\mathcal{D}$ as the environment, $\mathcal{O}_0$ as the protocol, and $\mathcal{O}_1$ as the simulator.

Remark 5.1.9 (Fat-tailed distributions). The sum $\sum_n n^{-c}$ is finite if and only if $c > 1$. Thus, we obtain a random variable X with $\Pr[X = n] \propto n^{-c}$. For $\gamma > 0$ we have $\mathbb{E}[X^\gamma] \propto \sum_n n^{-c+\gamma}$. If $c - \gamma \leq 1$, then $\mathbb{E}[X^\gamma] = \infty$. Moreover, $\Pr[X \geq k] \geq k^{-c}$, i.e. X has **fat tails**. In particular, for $c = 3$, $\mathbb{E}[X] < \infty$ but $\mathbb{E}[X^2] \propto \sum_n n^{-1} = \infty$, and $\Pr[X \geq \mathsf{poly}] \geq \frac{1}{\mathsf{poly}^3}$ for any poly.

Allowing a negligible deviation clearly separates perfect runtime distributions from their computational counterparts. Clearly, PPT is strictly contained in EPT. The separation of CPPT and CEPT follows from fat-tailed distributions. In Section 5.1.6.1 below, we separate CEPT from Levin's relaxations of EPT, denoted $\mathcal{LT}$, and Vadhan's relaxation [Gol10] of $\mathcal{LT}$, denoted $\mathcal{VT}$, which allows negligible deviation. In the following diagram, *strict* inclusions are denoted by arrows.

$$\begin{tikzcd}
\mathcal{PPT} \arrow[r] \arrow[d] & \mathcal{EPT} \arrow[r] \arrow[d] & \mathcal{LT} \arrow[d] \\
\mathcal{CPPT} \arrow[r] & \mathcal{CEPT} \arrow[r] & \mathcal{VT}
\end{tikzcd}$$

5.1.6.1. Levin's Relaxation and CEPT

We noted in Remark 5.1.9, that $\sum_{n=1}^{\infty} n^{-c} = \alpha_c < \infty$ for $c > 1$ gives rise to a distribution Z_c over $\mathbb{N}$ via normalizing the sum. Let $X = Z_2^3$. Then $\mathbb{E}[X] = \frac{1}{\alpha_c} \sum_{n=1}^{\infty} n = \infty$. Since Z_2 is fat-tailed, so is X. Let $Y_k = X|_{(\cdot \geq k^3) \mapsto 0}$. It follows immediately that $\mathbb{E}[Y_k] = \mathbb{E}[X|_{(\cdot \geq k^3)\mapsto 0}] \geq \frac{1}{\alpha_c} k^2$ for any $k \in \mathbb{N}$. Thus, for any superpolynomial cutoff K, we find $\mathbb{E}[Y_K] \geq \frac{1}{2\alpha_c} K^2$ is superpolynomial, and as a consequence, there is no superpolynomial cutoff which makes X EPT. (We interpret X as a constant family of runtimes, i.e. $X_\lambda = X$ for all λ.)

Formally, CEPT uses ν-quantile cutoffs (i.e. we may condition on an event $\mathcal{G}$ of overwhelming probability $1 - \nu$ that minimizes $\mathbb{E}[T \mid \mathcal{G}]$). For X, any ν-quantile cutoff for negligible ν induces some bound k which maximizes $\Pr[T \leq k] \geq \nu$. If k were polynomial, then (due to "fat tails") ν must also be polynomial. Hence, k must be superpolynomial, and consequently there is no negligible quantile cutoff which makes X EPT. All in all, the runtime distribution X is allowed by Levin's relaxation, but is not CEPT.

5.1.7. Structure of this Chapter

In Section 5.2, we clarify additional preliminaries, such as (non-)standard (notational) conventions, shorthands and terminology, and some basic concepts and results. In Section 5.3, we define CEPT and prove the characterization as well as generalizations and convenience lemmas. In Section 5.4, we introduce the standard reduction, relative efficiency and the hybrid lemma. In Section 5.5, we apply CEPT to zero-knowledge. We define (uniform complexity auxiliary input) zero-knowledge, and consider the example of $\mathrm{G3C}_{\mathrm{GK}}$ in detail. Then, we define sequential zero-knowledge and prove that it is implied by auxiliary input zero-knowledge. In Section 5.6, we define rewinding strategies, simple assumptions, and benign simulation, Moreover, we give a simple proof that benign simulators are (sequential) zero-knowledge. In Section 5.7, we sketch the application of CEPT to (uniform complexity) multiparty computation. In Section 5.8, we conclude and highlight some open questions.

In Appendix C.1, we give a detailed discussion on the effect of machine models and their (in)compatibility with expected time. Appendix C.2 contains supplementary definitions for commitment schemes. The remaining appendices contain further material and discussion. Appendix C.3 contains some simple but useful results and reminders for our general discussion of runtime classes. Appendix C.4 treats runtime more abstractly. It justifies the notion of "closed runtime classes" formally and demonstrates how most of our results extend to algebra-tailed runtime class. In Appendix C.5, we discuss asides for each chapter, and more. These provide clarifications, justify decisions, technical details, effects of variations in definitions, give (simple) examples, and so on. Finally, for completeness, we show in Appendix C.6 that our approach is applicable even if we follow the work of Katz and Lindell [KL08] much more closely, although at the expense of more convoluted proofs.

5.2. Preliminaries

In this section, we state some basic definitions and (non-)standard conventions used in this chapter. We also recall some basic definitions since dealing with low-level questions such as runtime is more sensitive to the concrete choices, although our final notions appear to be rather robust.

5.2.1. Notation and Basic Definitions

We denote the security parameter by λ; it is often suppressed. Similarly, we often speak of an object X, instead of a family of objects $(X_\lambda)_\lambda$ parameterized by λ. We always assume binary encoding of data, unless explicitly specified otherwise.[21] We write $X \sim Y$ if a random variable X is distributed as Y. For random variables X, Y over a set A We write $X|_{a\mapsto b}$ (resp. $X|_{S\mapsto b}$, resp. $X|_{\mathsf{pred}\mapsto b}$) for the random variable where a (resp. any a satisfying $a \in S$ resp. $\mathsf{pred}(a) = 1$) is mapped to b, and everything else unchanged, e.g. $X|_{\bot\mapsto 0}$ or $X|_{S\mapsto 0}$ or $X|_{\cdot\geq N\mapsto N}$.

For a countable set $\mathcal{S}$ and a function $\phi\colon \mathcal{S} \to \mathbb{R}$, let $\|\phi\|_p := (\sum_{x\in\mathcal{S}}|\phi(x)^p|)^{1/p}$ be the p-norms for $p \in [1,\dots,\infty]$. (Recall that $\|\phi\|_\infty := \sup_{x\in\mathcal{S}}|\phi(x)|$.) We define statistical distances $\Delta_p(\rho,\sigma) := \frac{1}{2}\|\rho-\sigma\|_p$ of distributions $\rho,\sigma\colon \mathcal{S} \to [0,1]$. Recall that $\Delta_1(\rho,\sigma) = \sup_{X\subseteq\Omega}|\rho(X)-\sigma(X)|$. We refer to the variational distance $\Delta(\,\cdot\,,\,\cdot\,) := \Delta_1(\cdot,\cdot)$ as the **statistical distance**.

We call $\rho_{\sup}(\mu/\nu) := \sup_x \frac{\mu(x)}{\nu(x)}$ (where $\frac{0}{0} := 0$) the **sup-ratio** of μ over ν; μ and ν may be arbitrary non-negative functions.

With poly, polylog, and negl we denote polynomial, polylogarithmic and negligible functions (in λ) respectively. Usually, we (implicitly) assume that poly, polylog, and negl are *monontone.* A function negl is (polynomially) negligible if $\lim_{\lambda\to\infty}\mathsf{poly}(\lambda)\mathsf{negl}(\lambda) = 0$ for every polynomial poly. In many definitions, we assume the existence of a negligible bound negl on some advantage $\varepsilon = \varepsilon(\lambda)$. We generally use "strict pointwise $\leq$" for bounds, e.g. $\varepsilon \leq \mathsf{negl}$ denotes $\forall\lambda\colon \varepsilon(\lambda) \leq \mathsf{negl}(\lambda)$. We avoid "eventually $\leq$", denoted $\varepsilon \leq_{\mathrm{ev}} \mathsf{negl}$ (defined via $\exists C\forall\lambda > C\colon \varepsilon(\lambda) \leq \mathsf{negl}(\lambda)$). If $\varepsilon \leq_{\mathrm{ev}} \mathsf{negl}$, then $\max\{\varepsilon(\lambda),\mathsf{negl}(\lambda)\} =: \nu(\lambda)$ is negligible and $\varepsilon \leq \nu$, hence this makes no difference in most situations. However, "$\leq$" behaves "more intuitively" than "$\leq_{\mathrm{ev}}$" in some sense.[22]

[21] In classical efficiency settings, unary encoded data is primarily used to model efficiency restrictions implicitly. We model these explicitly, and, due to a posteriori notions, efficiency depends only on λ anyway. It is irrelevant if λ is passed as binary or unary to the machines, hence we use binary encodings unless otherwise specified.

[22] When infinitely many functions are considered, $\leq$ and $\leq_{\mathrm{ev}}$ behave differently. For $\leq_{\mathrm{ev}}$, any countable set of negligible functions is $\leq_{\mathrm{ev}}$-dominated by some negl, c.f. [Bel02]. This is false for $\leq$. Indeed, $\leq_{\mathrm{ev}}$ behaves unintuitive. Consider a sum of a growing number (in λ) of negligible functions ν_i. It is well-known that $\mu(\lambda) := \sum_{i=1}^{\lambda}\nu_i(\lambda)$ need not be negligible, even if all ν_i are negligible. But if all ν_i are "strictly dominated" by some ν, i.e. $\nu_i \leq \nu$, then $\mu(\lambda) \leq \lambda\nu(\lambda)$ hence μ is negligible. However, if all ν_i are only "*eventually* dominated", i.e. $\nu_i \leq_{\mathrm{ev}} \nu$, then the standard counterexample ($\nu_i(j) = 1$ if $i = j$ and 0 else) shows that μ need not be negligible. Concretely, $\nu = 0$ *eventually* dominates all ν_i, yet $\mu(n) = 1 > 0 = n\nu(n)$. Due to this behaviour, we avoid "$\leq_{\mathrm{ev}}$".

5.2.2. Systems, Algorithms, Interaction and Machine Models

More detailed discussion of (unexplained) terms in this section are in Appendix C.1.

Machine Models. We fix some **admissible** machine model, which in particular implies that emulating a system of interacting machines has small overhead. The reader may assume a RAM model without much loss. In particular, polylogarithmic (emulation) overhead is acceptable in our setting, see. Appendix C.1.4.[23] Another irksome technicality are non-halting computations. One may follow [Gol10], and assume all algorithms halt after a finite number $n(\lambda)$ of steps. Instead, we deal with non-halting executions explicitly. For this, we define the symbol `nohalt` as the "output" of such a computation, and assume that any system which receives `nohalt` also outputs `nohalt`, if not specified otherwise.

Systems, Algorithms and Oracles. We always consider (induced) systems, which offer **interfaces** for (message-based) communication.[24] Input and output are modelled as interfaces as well. The security parameter λ is an implicit input interface of (almost) every system; a system is **closed** if its only interfaces are for λ and output, i.e. it is a "sampling algorithm" (which takes λ and samples some output). A **system** is a "mathematical" object, which defines (probabilistic) behaviour of the offered interfaces. An **algorithm** is given by *code*, a *finite*[25] string describing the behaviour and interfaces, and has a notion of runtime and randomness interface (e.g. random tape) which are imparted on it by the machine model. **Oracles** or **parties** are, unless stated otherwise, algorithms, which are only used via their interface. To emphasize availability of a certain oracle to some algorithm, we speak of **oracle algorithms**. A **timed** oracle offers an extended interface to its caller, which allows to bound the maximum time spent in an invocation (and return `timeout` if the allotted time is exceeded), and also returns the elapsed time of any invocation. Oracles also serve as a means to make **subroutine calls** explicit. A **timeful** oracle (or system) comes with some notion of *purported elapsed runtime*. For consistency, the purported elapsed runtime is always at least the answer length of an invocation, and this is usually also the runtime notion of interest. Timeful oracles (or systems) are used as convenience abstractions to specify and analyze unconditional properties. Timed timeful oracles are defined in the obvious way.

Interaction. It will always be clear from the context how interfaces are used or connected. Interactivity is implicit, and implied by open interfaces. Let $\mathsf{A}_1, \mathsf{A}_2$ be a algorithms (or more generally, systems). For connecting A_1 and A_2, i.e. interaction, with (fixed) inputs x, y, z, we write $\langle \mathsf{A}_1(x,z), \mathsf{A}_2(y,z)\rangle$. The result is another algorithm (or system), where we write $\mathrm{out}_{\mathsf{A}_i}\langle \mathsf{A}_1, \mathsf{A}_2\rangle$ for the output (interface) of A_i for $i = 1, 2$. We write $\mathsf{A}^{\mathcal{O}}$ for an algorithm (or system) A, with access to an oracle $\mathcal{O}$ (where $\mathcal{O}$ may be a subroutine, e.g. a commitment scheme). This notation emphasizes, that the output of the system is that of A. Otherwise, the system is equivalent to $\langle \mathsf{A}, \mathcal{O}\rangle$, or even $\mathcal{O}^{\mathsf{A}}$. We view interaction, oracle, and subroutine calls as essentially identical and use the notation interchangeably if no confusion arises.

Black-box rewinding (bb-rw) access to an algorithm A (or timeful system) means access to an oracle $\mathsf{bbrw}(\mathsf{A})$ emulating A with fresh but fixed randomness, which allows to feed A messages and rewind it

[23] More precisely, CEPT is robust w.r.t. polylogarithmic overhead, due to virtuality. For robustness of EPT, an additional strict a priori runtime bound is needed, e.g. $2^{\mathrm{poly}(\lambda)}$ works.

[24] We use an ad-hoc definition of system. A compatible, precise notion was recently (concurrently) introduced in [LM20].

[25] Non-uniform notions deviate here and allow infinite descriptions.

to any visited state. For notational simplicity, we treat $\mathsf{bbrw}(\mathsf{A})$ like a $\mathsf{NextMsg}_{\mathsf{A}}$ function, which upon a query $query = (m_1, \dots, m_n)$ returns the result of A when given m_i as its i-th message. The query $(m_1, \dots, m_n)$ is viewed as a *logical handle* $(m_1, \dots, m_{n-1})$ to a previously visited state, and a message m_n to A when in that state. Implementations of $\mathsf{bbrw}(\mathsf{A})$ use *short* handles, say a counter. A **timed** bb-rw oracle truncates and returns the elapsed runtime of its emulated program. By abuse of notation, we often write B^{A} instead of $\mathsf{B}^{\mathsf{bbrw}(\mathsf{A})}$ if it is clear that B has bb-rw access to A.

Remark 5.2.1 (Efficient implementations). Access to $\mathsf{NextMsg}_{\mathsf{A}}$ and $\mathsf{bbrw}(\mathsf{A})$ is "*logically equivalent*", yet, the efficiency characteristics differ vastly. For expected time, this is a critical point. We encounter such issues also in other situations, and will offer a brief warning but proceed with the usual notation. Using more efficient "logically equivalent" implementations solves such problems. See Appendix C.1.3.

5.2.3. Input Generation: Conventions and Shorthands

In *non*-uniform complexity settings, it is possible to quantify over all inputs to a protocol universally. In uniform complexity settings [Gol93], these inputs must be *efficiently samplable*. For this, we use efficient algorithm, usually denoted $\mathcal{I}$, called the **input generator**. For non-uniform security, $\mathcal{I}$ is non-uninform, i.e. has tape-like access to an (unbounded) non-uniform advice string $advc_\lambda$. This deviates from standard definitions [Gol01] slightly by allowing input *distributions*.

Notation 5.2.2 (Shorthand expressions for composing systems). Let P, V^* be two (interacting) parties and let $\mathcal{I}$ be an input generator. We use the shorthand notation $\langle \mathsf{P}, \mathsf{V}^* \rangle_{\mathcal{I}}$ for the system resp. interaction of $\langle \mathsf{P}, \mathsf{V} \rangle$ completed with $\mathcal{I}$, where it is either clear how to connect the interfaces or it is explicitly described. We also say: "Let $\mathrm{out}_{\mathsf{V}^*}\langle \mathsf{P}(\mathbb{x}, \mathbb{w}), \mathsf{V}^*(\mathbb{x}, aux) \rangle$, where $(\mathbb{x}, \mathbb{w}, aux) \xleftarrow{\$} \mathcal{I}(\lambda)$."

What we mean by this is: Consider the system obtained by composing $\mathcal{I}$, P and V^* as indicated, that is, the system which first runs $\mathcal{I}$ to obtain $(\mathbb{x}, \mathbb{w}, aux)$, then passes $(\mathbb{x}, \mathbb{w})$ to P as input, and passes $(\mathbb{x}, aux)$ to V^*, and then runs P and V^* (i.e. letting them interact). Of this composed system, take and return the output of V^*.

Note that we do *not* mean to quantify over all inputs $(\mathbb{x}, \mathbb{w}, aux)$ which $\mathcal{I}$ may produce, except if made explicit, e.g. by stating "*for all* $(\mathbb{x}, \mathbb{w}, aux) \leftarrow \mathcal{I}$" or more precisely "*for all* $(\mathbb{x}, \mathbb{w}, aux) \in \mathrm{supp}(\mathcal{I})$". Since we almost exclusively consider closed systems, and fixed inputs make little sense in a uniform asymptotic setting, no confusion should arise.

5.2.4. Preliminary Remarks on Runtime

An abstract treatment of runtime is in Appendix C.4, and meant for the inclined reader only. This section contains all essential definitions for Section 5.3 and later sections, which only deal with polynomial times, namely PPT, EPT, CPPT and CEPT.

For an oracle algorithm A, we write $\mathsf{time}_{\mathsf{A}}(\mathsf{A}^{\mathcal{O}})$ for the time spent in A (called **oracle-excluded time**), $\mathsf{time}_{\mathcal{O}}(\mathsf{A}^{\mathcal{O}})$ for the time spent in $\mathcal{O}$, and $\mathsf{time}_{\mathsf{A}+\mathcal{O}}(\mathsf{A}^{\mathcal{O}})$ for the time spent in both (called **oracle-included time**). This notation extends naturally to interaction and composite systems built from interacting machines. Note that $T = \mathsf{time}_{\mathsf{A}}(\mathsf{A}^{\mathcal{O}})$ is a *random variable*, or more precisely, a sequence of random variables T_λ parameterized by λ. We assume that that runtimes sum up, i.e. $\mathsf{time}_{\mathsf{A}}(\mathsf{A}^{\mathcal{O}}) + \mathsf{time}_{\mathcal{O}}(\mathsf{A}^{\mathcal{O}}) = \mathsf{time}_{\mathsf{A}+\mathcal{O}}(\mathsf{A}^{\mathcal{O}})$, *as dependent random variables.*

Definition 5.2.3. A **runtime (distribution)** T is a family of random variables (resp. distributions) over $\mathbb{N}_0$ parameterized by the security parameter λ. We (only) view a runtime as a random variable $T_\lambda \colon \Omega_\lambda \to \mathbb{N}_0$, when stochastic dependency is relevant.

Definition 5.2.4. A **runtime class** $\mathcal{T}$ is a set of runtime distributions.[26] A (sampling) algorithm A is $\mathcal{T}$**-time** if $\mathsf{time}_{\mathsf{A}}(\mathsf{A}) \in \mathcal{T}$, more explicitly, $T_\lambda = \mathsf{time}_{\mathsf{A}}(\mathsf{A}(\lambda))$ is in $\mathcal{T}$.

Example 5.2.5. The runtime classes $\mathcal{PPT}$ and $\mathcal{EPT}$ of strict polynomial time (PPT) and expected polynomial time (EPT) are defined in the obvious way, i.e.: $T \in \mathcal{PPT}$ (resp. $T \in \mathcal{EPT}$) if there exists a polynomial poly such that $\Pr[T_\lambda > \mathsf{poly}(\lambda)] = 0$ (resp. $\mathbb{E}[T_\lambda] \le \mathsf{poly}(\lambda)$).

Our central tool for dealing with expected time is truncation. Also recall that timed oracles abstract the ability to truncate executions.

Definition 5.2.6 (Runtime truncation). Let A be an algorithm. We define $\mathsf{A}^{\le N}$ as the algorithm which executes A up to N steps, and then returns A's output. If A did not finish in time, $\mathsf{A}^{\le N}$ returns `timeout`.

A priori Time, a posteriori Time, and Designated Adversaries. In any closed system, every component has an associated random variable, describing the time spent in it. We only consider such runtimes (most often, the total runtime). Hence, efficiency depends only on λ, since closed systems have no (other) input. In particular, we do not assign a stand-alone notions of efficiency or runtime to a non-closed system, e.g. an algorithm A which needs inputs (besides λ), resp. oracle access, resp. communication partners. The exception to the rule are *a priori PPT resp. EPT* algorithms A, for which there is a bound poly such that $\mathsf{time}_{\mathsf{A}}(\dots) \le \mathsf{poly}$ resp. $\mathbb{E}[\mathsf{time}_{\mathsf{A}}(\dots)] \le \mathsf{poly}$ for any choice of inputs, oracles, and communication partners.[27]

A posteriori efficiency of algorithms (or systems) considers them in a complete context, i.e. as part of a closed system. Let A be an algorithm and $\mathcal{E}$ be an environment such that $\langle \mathcal{E}, \mathsf{A} \rangle$ is a closed system. For **a posteriori** time, there are two sensible definitions: We can call A a posteriori PPT (resp. EPT, …) w.r.t. $\mathcal{E}$, if $\mathsf{time}_{\mathsf{A}}(\langle \mathcal{E}, \mathsf{A} \rangle)$ is PPT (resp. EPT, …), or if $\mathsf{time}_{\mathcal{E}+\mathsf{A}}(\langle \mathcal{E}, \mathsf{A} \rangle)$ is PPT (resp. EPT, …). We generally use the latter, but are always explicit about it. Applied to security notions, we get **designated adversaries**, which need only be *efficient for the protocol they are designed to attack*, see [Fei90] or [KL08; Gol10].

5.2.5. Probability Theory

By $\mathsf{Dists}(X)$ we denote the space of probability distributions on X. The underlying probability space for random variables is usually denoted by Ω, the associated σ-algebra is always left implicit. We neglect measurability questions because they do not pose any problems and are merely trivial technical overhead, see Appendix C.5.8 for a brief discussion.

We allow product extension of Ω to suit our needs, say extending to $\Omega' = \Omega \times \Sigma$ with Bernoulli distribution $\mathsf{Ber}(\frac{1}{3})$ on $\Sigma = \{0, 1\}$. Random variables over Ω are lifted implicitly and we again write Ω instead of Ω'. Let $\mathbb{N}_0 \cup \{\infty, \mathtt{timeout}\}$ be totally ordered via $n < \infty < \mathtt{timeout}$ for all $n \in \mathbb{N}_0$. For $X \colon \Omega \to \mathbb{R}$, if $\Pr[X > c] < \mathsf{tail}(c)$, we call tail a **tail bound**. For families $X_\lambda \colon \Omega \to \mathbb{R}$, we sometimes

[26] For our general treatment of runtimes, we use a more restrictive definition, c.f. Appendix C.4.3.

[27] By definition, a priori PPT is the essentially same as a priori PPT in any interaction of [KL08; Gol10], but in our setting where only the security parameter grants runtime. Note that "classical" PPT algorithms are not a priori PPT in our sense, since their runtime bound depends on the input size, while ours are fixed by λ alone. We can mitigate this discrepancy by size-guarding (see Appendix C.5.4.3).

sloppily call t_λ a *tail bound* for c_λ if $\Pr[X_\lambda > c_\lambda] < t_\lambda$. We denote the cumulative density function (CDF) of X by $\mathsf{CDF}_X(c) = \Pr[X \le c]$ and let $\overline{\mathsf{CDF}}_X(\,\cdot\,) := 1 - \mathsf{CDF}_X(\,\cdot\,) = \Pr[X > \,\cdot\,]$.

For convenience, we use a relaxation of stochastic domination.

Definition 5.2.7 (Domination with slack). Let $X, Y\colon \Omega \to \mathcal{S}$ be random variables and $\mathcal{S}$ be a totally ordered set (usually $\mathcal{S} = \mathbb{R} \cup \{\mathtt{timeout}\}$). Let $L \ge 1$. We say Y **dominates** X **with slack** L (in distribution), if $\overline{\mathsf{CDF}}_X \le L \cdot \overline{\mathsf{CDF}}_Y$, that is, if

$$\forall c \in \mathcal{S}: \qquad \Pr[X > c] \le L \cdot \Pr[Y > c].$$

We denote this by $X \overset{d}{\le}_L Y$. If $L = 1$, we write $X \overset{d}{\le} Y$. We use the same notation for families of random variables, i.e. we write $X \overset{d}{\le} Y$ and mean $X_\lambda \overset{d}{\le} Y_\lambda$ for all λ.

Instead of truncating runtimes in the domain, we often "truncate" in the probability space.

Definition 5.2.8 (ν-quantile cutoff). Let T be a distribution on $\mathbb{N}_0 \cup \{\infty\}$ and $\nu > 0$. Suppose that $\Pr[T = \infty] \le \nu$.[28] The (exact) ν-**quantile (cutoff)** T^ν is following distribution on $\mathbb{N}_0 \cup \mathtt{timeout}$. Let $\mathsf{CDF}_T(\,\cdot\,)\colon \mathbb{N}_0 \cup \{\infty\} \to [0,1]$ be the CDF of T. Then $\mathsf{CDF}_{T^\nu}(\,\cdot\,)\colon \mathbb{N}_0 \cup \mathtt{timeout} \to [0,1]$ is defined by $\mathsf{CDF}_{T^\nu}(n) = \min\{1-\nu, \mathsf{CDF}_T(n)\}$ for $n \in \mathbb{N}$, and $\mathsf{CDF}_{T^\nu}(\infty) = \lim_{n\to\infty} \min\{1-\nu, \mathsf{CDF}_T(n)\}$, hence $\Pr[T^\nu = \infty] = 0$, and $\mathsf{CDF}_{T^\nu}(\mathtt{timeout}) = 1$,

An exact ν-quantile cutoff for a random variable $T\colon \Omega \to \mathbb{N}_0 \cup \{\infty\}$ can be constructed by: First pick $N = \inf\{n \mid \Pr[T > n] \le \nu\}$. If $\Pr[T > N] =: \nu'$ equals ν, let $T^\nu := T|_{\cdot > N \mapsto \mathtt{timeout}}$. Else, pick a (measurable) subset of $A = \{\omega \in \Omega \mid T(\omega) = N\}$ of probability $\nu - \nu'$, and let $T^\nu := T|_{A \mapsto \mathtt{timeout}}$. If necessary, modify Ω. So we assume w.l.o.g. that there is such a set of events. An *approximate ν-quantile cutoff* with *error* δ is an exact ν'-quantile cutoff, where $\nu \le \nu' \le \nu + \delta$.

In case of discrete distributions, one can find a unique maximal (measurable) subset A (e.g. minimal by lexicographic order), and a unique atomic event which may have to be split between N and $\mathtt{timeout}$. By modifying Ω to $\Omega \times \{0,1\}^n$, an approximate cutoff with error at most to 2^{-n} is possible. Using $\Omega \times \mathsf{Ber}(\nu - \nu')$, exact cutoffs are possible.

Remark 5.2.9 (Equal-unless). If $X, Y\colon \Omega \to \mathcal{S}$ are random variables and coincide (as functions), except for an event $\mathcal{E} \subseteq \Omega$, then X and Y are **(pointwise) equal unless** $\mathcal{E}$. Typically, $\mathcal{E} = \{\omega \mid Y(\omega) = \mathtt{bad}\}$ (for some symbol $\mathtt{bad}$), and we say X equals Y unless $\mathtt{bad}$ happens. We also say X and Y *coincide unless* (or *agree except*) if $\mathtt{bad}$ happens. The definition naturally extends to oracles and systems.

Remark 5.2.10 (Truncation of values vs. quantiles). Consider random variables X, Y over $\mathbb{R}$ with $X \overset{d}{\le}_L Y$ (for some $L \ge 1$). As seen in Lemma C.3.7, quantile-truncation preserves domination even if we additionally condition on $\neg\mathtt{timeout}$. Truncating in the domain does *not* preserve domination if we additionally condition on $\neg\mathtt{timeout}$. For example, over $\{1,2,3,4\}$ consider the probability vectors $p_X \mathrel{\widehat{=}} (\beta, 0, 1-\beta, 0)$ and $p_Y \mathrel{\widehat{=}} (0, \alpha, 0, 1-\alpha)$. Truncating X, Y at 3 and conditioning on $\neg\mathtt{timeout}$ yields X', Y' with $p_{X'} = p_X$ and $p_{Y'} = (0,1,0)$, and thus $X' \overset{d}{\nleq}_L Y'$, even for $L = 1$.

[28] It is straightforward to deal with general $\nu \ge 0$. But distributions S over $\mathbb{N}_0 \cup \{\infty\} \cup \mathtt{timeout}$ with $\Pr[S = \infty] > 0$ are not particularly useful for us.

5.2.6. Indistinguishability and Oracle-Related Notions

We define (oracle-)indistinguishability, repeated trials, and query sequences.

5.2.6.1. Oracle-Indistinguishability

The (in)distinguishability of oracles (or systems) is a folklore abstraction. "Bit-guessing" experiments, such as indistinguishability of distributions, and more generally game-based security notions can be straightforwardly rephrased as an oracle pair, see Appendix C.5.1.3. Depending on the oracles (or systems) and their interfaces, distinguishing can encompass (adversarial) input generation, protocol runs, and more. For example, an oracle may present an IND-CPA game for public key encryption, or it may present the distinguisher with a concurrent zero-knowledge setting.

Definition 5.2.11 (Oracle-indistinguishability). Let $\mathcal{O}_0$ and $\mathcal{O}_1$ be (not necessarily computable) oracles with identical interfaces. A distinguisher $\mathcal{D}$ is a system which connects to all interfaces or $\mathcal{O}_0$, $\mathcal{O}_1$, resulting in a closed system $\mathcal{D}^{\mathcal{O}_b}$. The **(standard) distinguishing advantage** of $\mathcal{D}$ is defined by

$$\mathrm{Adv}^{\mathrm{dist}}_{\mathcal{D},\mathcal{O}_0,\mathcal{O}_1}(\lambda) = |\Pr[\mathcal{D}^{\mathcal{O}_1(\lambda)}(\lambda) = 1] - \Pr[\mathcal{D}^{\mathcal{O}_0(\lambda)}(\lambda) = 1]|.$$

By abuse of notation, we sometimes abbreviate $\mathrm{Adv}^{\mathrm{dist}}_{\mathcal{D},\mathcal{O}_0,\mathcal{O}_1}$ by $\mathrm{Adv}^{\mathrm{dist}}_{\mathcal{D},\mathcal{O}}$.

Let $\mathcal{T}$ be a runtime class. Then $\mathcal{O}_0$ and $\mathcal{O}_1$ are **computationally (standard) indistinguishable in $\mathcal{T}$-time**, written $\mathcal{O}_0 \overset{c}{\approx}_{\mathcal{T}} \mathcal{O}_1$ if for any $\mathcal{T}$-time distinguisher $\mathcal{D}$, i.e. $\mathsf{time}_{\mathcal{D}}(\mathcal{D}^{\mathcal{O}_b(\lambda)}(\lambda)) \in \mathcal{T}$ (for $b = 0, 1$),[29] there is some negligible negl such that $\mathrm{Adv}^{\mathrm{dist}}_{\mathcal{D},\mathcal{O}}(\lambda) \leq \mathsf{negl}$. We define **statistical** $\mathcal{T}$-query indistinguishability by counting only oracle-queries as runtime.

Perfect indistinguishability is special, and we reserve the notation "$\equiv$" for it.

Definition 5.2.12. Oracles $\mathcal{O}_0$, $\mathcal{O}_1$ (or systems, or algorithms), for which all (unbounded) distinguishers have advantage 0 are called **perfectly indistinguishable**. We also write $\mathcal{O}_0 \equiv \mathcal{O}_1$ to emphasize this.

Remark 5.2.13 (Indistinguishability of distributions). Indistinguishability of distributions X and Y (under repeated samples) is defined in the natural compatible way, namely via oracles $\mathcal{O}_X$ and $\mathcal{O}_Y$ which output a single (a fresh) sample of X resp. Y (for each query).

5.2.6.2. Repeated Trials

It is useful to make repeated oracle access explicit.

Definition 5.2.14 (Repeated oracle access). Let $\mathcal{O}$ be an oracle. We denote by $\mathsf{rep}(\mathcal{O})$ an oracle which offers repeated access to *independent instances* of $\mathcal{O}$. For example, $\mathsf{rep}(\mathcal{O})$ may implement this by expecting message tuples (i, m) of oracle index i and query m, and a special message which starts a new independent copy of $\mathcal{O}$, increasing the maximal admissible index i by 1. We denote by $\mathsf{rep}_q(\mathcal{O})$ an oracle which limits access to a total of at most q instances of $\mathcal{O}$. (Effectively, the admissible indices are $1, \dots, q$. Also observe that $\mathsf{rep}(\mathcal{O}) = \mathsf{rep}_\infty(\mathcal{O})$.)

[29] This is equivalent to being efficient in the respective distinguishing experiment.

Definition 5.2.15 (Indistinguishability under repeated trials). Let $\mathcal{O}_0$ and $\mathcal{O}_1$ be two oracles. We say $\mathcal{O}_0$ and $\mathcal{O}_1$ are $\mathcal{T}$**-time computationally indistinguishable under** q **(repeated) trials**, if $\mathrm{rep}_q(\mathcal{O}_0)$ and $\mathrm{rep}_q(\mathcal{O}_1)$ are $\mathcal{T}$-time computationally indistinguishable. We say $\mathcal{O}_0$ and $\mathcal{O}_1$ are $\mathcal{T}$**-time indistinguishable under (unbounded many) repeated trials**, if they are $\mathcal{T}$-time indistinguishable for $q = \infty$ repeated trials. The definition for $\mathcal{T}$**-query statistically indistinguishable** is analogous.

5.2.6.3. Query-Sequences

We use following definition and notation for the sequence of queries made by an algorithm to its oracle.

Definition 5.2.16 (Query-sequence). Let $\mathsf{A}^{\mathcal{O}}$ be an oracle algorithm. The **query-sequence** $\mathrm{qseq}_{\mathcal{O}}(\mathsf{A}^{\mathcal{O}}(x))$ is the (distribution of the) sequence of queries made by A to $\mathcal{O}$. We view $\mathrm{qseq}_{\mathcal{O}}(\mathsf{A}^{\mathcal{O}}(x))$ as an oracle, which grants lazy (tape-like) access to the queries.

5.3. Computationally Expected Polynomial Time

In this section, we define computationally expected polynomial time (CEPT), briefly recap the general results of Appendix C.4 for polynomial runtime classes, and have a first glimpse of the behaviour of CEPT. The inclined reader may wish to continue with Appendix C.4 instead; it deals with runtime classes in more generality.

5.3.1. A Brief Recap

5.3.1.1. Virtually Expected Time

We are interested in properties, which need only hold with overwhelming probability. We formalize this for the expectation of non-negative random variables as follows.

Definition 5.3.1 (Virtual expectation). Let $X\colon \Omega \longrightarrow \mathbb{R}_{\geq 0} \cup \{\infty\}$ Let $\varepsilon > 0$. We say X has ε**-virtual expectation (bounded by)** t if

$$\exists \mathcal{G} \subseteq \Omega\colon \Pr[\mathcal{G}] \geq 1 - \varepsilon \ \wedge\ \mathbb{E}[X \mid \mathcal{G}] \leq t$$

We extend this to families by requiring it to hold component-wise. Moreover, we say a runtime T is ε**-virtually** t**-time** if T has ε-virtual expectation bounded by t. We abbreviate this as **virtually expected** (t, ε)**-time** and call ε the **virtuality** of time (t, ε).

Virtual properties have a "probably approximately" flavour. They are closely related to "ε-smooth properties", such as ε-smooth min-entropy, which smudge over statistically close random variables (instead of conditioning).[30] Virtual properties must behave well under restriction (up to a certain extent).

Lemma 5.3.2. *Let* $X\colon \Omega \longrightarrow \mathbb{R}_{\geq 0}$ *be a random variable and* $\mathbb{E}[X] = t$. *Then any restriction of* X *to an event* $\mathcal{G}$ *of measure* $1 - \varepsilon$ *implies* $\mathbb{E}[X \mid \mathcal{G}] \leq (1 - \varepsilon)^{-1} t$.

[30] We borrowed the terminology of virtual properties from group theory.

The upshot of Lemma 5.3.2 is that, if we condition on an *overwhelming* (in fact, noticeable) event $\mathcal{G}$, polynomially bounded expectation is preserved. Also, consecutive restrictions of Ω are unproblematic.

5.3.1.2. Triple-Oracle Indistinguishability

Using *triple-oracle indistinguishability*, instead of standard indistinguishability, for (runtime) distributions abstracts technical details and prevents technical problems. Recall that we always use *binary* encodings, and this includes runtime oracles (even though unary encodings work there without change).

Definition 5.3.3. A **triple-oracle distinguisher** $\mathcal{D}$ for distributions X_0, X_1, receives access to three oracles $\mathcal{O}_0$, $\mathcal{O}_1$ and $\mathcal{O}_b^*$, which sample according to some distributions X_0, X_1, and X_b. The distinguishing advantage is $\mathrm{Adv}^{\text{3-dist}}_{\mathcal{D},\mathcal{O}_0,\mathcal{O}_1} = |\Pr[\mathcal{D}^{\mathcal{O}_0,\mathcal{O}_1,\mathcal{O}_1^*}(\lambda) = 1] - \Pr[\mathcal{D}^{\mathcal{O}_0,\mathcal{O}_1,\mathcal{O}_0^*}(\lambda) = 1]|$.

Two runtime distributions T, S are **computationally $\mathcal{T}$-time triple-oracle indistinguishable**, denoted by $T \overset{c}{\approx}_{\mathcal{T}} S$, if any $\mathcal{T}$-time distinguisher has advantage $o(1)$. If $\mathcal{T}$ contains $\mathcal{PPT}$, then (by amplification) any distinguisher has negligible advantage. For statistical triple-oracle indistinguishability, we only count oracle queries as steps (and often explicitly speak of statistical $\mathcal{T}$-query distinguishers)[31] and write $T \overset{s}{\approx}_{\mathcal{T}} S$.

A runtime class $\mathcal{T}$ is **computationally closed** if for all runtimes S, if there exists some $T \in \mathcal{T}$ such that $T \overset{c}{\approx}_{\mathcal{T}} S$, then $S \in \mathcal{T}$. **Statistically closed** is defined analogously.

In the definition, we sketched our approach for general runtime classes (namely requiring $o(1)$ advantage bound, see Appendix C.4). This definition applies to runtime classes from other algebras, such as polylog or quasi-polynomial time, and implicitly uses the notion of negligible function for these algebras. The use of tail bounds as our proof technique seems limited to the setting, where "advantage" and "time" algebras coincide. From now on, we specialize to the polynomial setting, where amplification enforces (poly-)negligible advantage.

Triple-oracle distinguishing should be interpreted as distinguishing with repeated samples, plus sampling access to the distributions X_0, X_1. It allows for quite modular reductions, as we see now.

Remark 5.3.4 (Standard and triple-oracle indistinguishability). To clearly distinguish triple-oracle and "normal" indistinguishability, we call the latter *standard* when in doubt. We use (and defined) triple-oracle indistinguishability only for (runtime) distributions, not for general oracles.

5.3.2. Characterizing CEPT

We begin with the fundamental definition of this section.

Definition 5.3.5 (CEPT and CPPT). The runtime class $\mathcal{CEPT}$ of **computationally expected polynomial time** contains all runtimes which are (triple-oracle) $\mathcal{PPT}$-time indistinguishable from expected polynomial time. In other words: A runtime T is CEPT if there is an EPT $\widetilde{T}$, such that T and $\widetilde{T}$ are triple-oracle $\mathcal{PPT}$-time indistinguishable, i.e. $T \overset{c}{\approx} \widetilde{T}$.

[31] We *never* consider unbounded queries for statistical triple-oracle distinguishing, as this trivially coincides with perfect indistinguishability.

Computationally (strict) probabilistic polynomial time is defined analogously and denoted $\mathcal{CPPT}$.

Now, we turn towards the characterization of CEPT. We start with a few simple lemmata. Their central technique is to approximate probability distributions with suitable precision, and then use this information for distinguishing.

Lemma 5.3.6. *Suppose S and T are runtimes and $T \in \mathcal{CEPT}$. Then statistical $\mathcal{CEPT}$-query and computational $\mathcal{CEPT}$-time triple-oracle indistinguishability coincide, i.e. $S \overset{c}{\approx}_{\mathcal{PPT}} T \iff S \overset{s}{\approx}_{\mathcal{PPT}} T$. Moreover, a priori PPT distinguishers are sufficient.*

Proof sketch. It is clear that statistical indistinguishability implies computational indistinguishability. Thus, we concentrate on the converse. For $T \in \mathcal{CEPT}$ there exists, by definition, some $\widetilde{T} \in \mathcal{EPT}$ such that $T \overset{c}{\approx} \widetilde{T}$ (triple-oracle *computational* indistinguishability). Hence, for any efficiently computable $N = N(\lambda)$, we have $|\Pr[T > N] - \Pr[\widetilde{T} > N]| \leq \mathsf{negl}$.

We show that T and $\widetilde{T}$ are *statistically* triple-oracle indistinguishable as well. Assume the statistical distance $\Delta(T, \widetilde{T})$ is at least $\delta = \frac{1}{\mathsf{poly}_0}$ infinitely often. Note that $\Pr[\widetilde{T} > N] \leq \frac{\mathsf{poly}_1}{N}$, where $\mathbb{E}[\widetilde{T}] \leq \mathsf{poly}_1$. Thus, by truncating $T, \widetilde{T}$ after, say $N = 4\mathsf{poly}_0\mathsf{poly}_1$, we know that $T^{\leq N}$ and $\widetilde{T}^{\leq N}$ are distributions with *polynomial support in* $\{0, \dots, N\}$ and *non-negligible statistical distance* $\frac{\delta}{4}$ infinitely often. Since we have (repeated) sample access to $T, \widetilde{T}$ and the challenge runtime, we can approximate the probability distributions (by the empirical probabilities) up to any $\frac{1}{\mathsf{poly}}$ precision in polynomial time, see Appendix C.3.4. Consequently, we can construct a (computational) PPT distinguisher if T and $\widetilde{T}$ are not statistically $\mathcal{PPT}$-query indistinguishable.

The described statistical-to-computational distinguisher works for T and S as well. Let $\delta = \Delta(T, S)$. Since $T \in \mathcal{CEPT}$, there is a suitable tail bound N with $\Delta(T, T^{\leq N}) \leq \frac{\delta}{4}$. It is easy to see that $\Delta(T^{\leq N}, S^{\leq N}) \geq \frac{\delta}{4}$.[32] If $\delta \geq \frac{1}{\mathsf{poly}}$ infinitely often, then there is a suitable polynomial N, such $\Delta(T^{\leq N}, S^{\leq N}) \geq \frac{\delta}{4}$ infinitely often. Thus, we are in the same setting as before, and can distinguish by approximation. Lastly, note that the distinguisher we constructed is a priori PPT. □

The proof of Lemma 5.3.6 also shows closedness of $\mathcal{CEPT}$.

Corollary 5.3.7. *Let T, S be two runtimes and $T \in \mathcal{CEPT}$. Then $S \overset{c/s}{\approx}_{\mathcal{PPT}} T \iff S \overset{c/s}{\approx}_{\mathcal{CEPT}} T$, i.e. statistical $\mathcal{PPT}$-query and computational $\mathcal{PPT}$-time triple-oracle indistinguishability coincide.*

Thus, we have shown that all relations $\overset{m}{\approx}_{\mathcal{T}}$ for $m \in \{c, s\}$, $\mathcal{T} \in \{\mathcal{PPT}, \mathcal{CEPT}\}$ coincide. For concrete applications, we want to use standard indistinguishability instead of triple-oracle indistinguishability whenever possible.

Lemma 5.3.8. *Let T and S be runtimes induced by algorithms* A, B, *and suppose $T \in \mathcal{CEPT}$. Then triple-oracle and standard $\mathcal{PPT}$-time indistinguishability coincide, i.e. $S \overset{c/s}{\approx}_{\mathcal{PPT}} T \iff S \overset{c/s}{\approx}_{\mathcal{PPT}} T$.*

[32] Intuitively, either `timeout` accumulates a difference in probability of $\frac{\delta}{4}$, or a difference of $\frac{\delta}{4}$ in probability is present on $\{0, \dots, N\}$, see Corollary C.3.5.

Proof sketch. Suppose T and S are triple-oracle distinguishable with advantage at least $\delta = \frac{1}{\mathsf{poly}_0}$ infinitely often. The distinguisher $\mathcal{D}'$ from the proof of Lemma 5.3.6 is a priori PPT with advantage $\frac{\delta}{4}$ infinitely often. Moreover, $\mathcal{D}'$ *truncates all samples* at *polynomial* N, i.e. $\mathcal{D}$ actually distinguishes $T^{\leq N}$ and $S^{\leq N}$. These truncated runtime distributions can be *sampled via emulation* in strict polynomial time. By sampling via emulation and a hybrid argument, we find an a priori PPT distinguisher $\mathcal{D}$ with advantage at least $\frac{\delta}{4N}$ infinitely often. □

We stress that to efficiently distinguish two induced runtimes, it is sufficient that *one* of the two algorithms is efficient.[33]

Putting things together yields following convenient characterization of CEPT and CPPT:

Corollary 5.3.9 (Characterization of CEPT). *Let T be a runtime. The following conditions are equivalent:*

0. *T is in $\mathcal{CEPT}$.*
1. *T is $\mathcal{PPT}$-time triple-oracle computationally indistinguishable from some $\widetilde{T} \in \mathcal{EPT}$.*
2. *T is $\mathcal{PPT}$-query triple-oracle statistically indistinguishable from some $\widetilde{T} \in \mathcal{EPT}$.*
3. *T is virtually expected polynomial time. Explicitly: There is a negligible function* negl*, an event $\mathcal{G}$ with* $\Pr[\mathcal{G}] \geq 1 - \mathsf{negl}$*, and a polynomial* poly*, such that* $\mathbb{E}[T_\lambda \mid \mathcal{G}] \leq \mathsf{poly}(\lambda)$.

Furthermore, $T \in \mathcal{CEPT}$ satisfies the following tail bound

$$\Pr[T_\lambda > N] \leq \frac{\mathsf{poly}(\lambda)}{N} + \mathsf{negl}(\lambda)$$

for poly *and* negl *as in 3. Consequently, $\mathcal{CEPT}$ distinguishers are not more powerful than $\mathcal{PPT}$ distinguishers. In particular, $\mathcal{CEPT}$ is a closed runtime class. (In fact, it is the closure of $\mathcal{EPT}$.)*

For induced runtimes $T = \mathsf{time}_{\mathsf{A}}(\mathsf{A})$, $S = \mathsf{time}_{\mathsf{B}}(\mathsf{B})$, *where $T \in \mathcal{CEPT}$, and S is arbitrary, computational $\mathcal{CEPT}$-time (resp. statistical $\mathcal{CEPT}$-query) triple-oracle indistinguishability and standard computational (resp. statistical) indistinguishability coincide.*

The analogous characterization and properties hold for CPPT.

The essence of Corollary 5.3.9 is the equivalence of items 1 and 3. The former is easy to prove, as it is follows by reductions to indistinguishability assumptions. The latter is easy to use, as it guarantees that, after ignoring a negligible set of bad events, one can work with perfect EPT.

Proof sketch of Corollary 5.3.9. Equivalence of items 1 and 2 follows from Lemma 5.3.6. Now, we show that 2 implies 3. For our triple-oracle notion, being statistically indistinguishable implies statistically closeness, as one can see by approximating the probability distribution, as in Lemma 5.3.6. By assumption, there exists some $\widetilde{T} \in \mathcal{EPT}$ with $\Delta(T, \widetilde{T}) \leq \nu$ negligible. Let T^ν be the ν-quantile of T. Clearly, $T^\varepsilon|_{\mathtt{timeout}\mapsto 0}$ minimizes the value of $\mathbb{E}[S]$ under the constraint that S is a non-negative random variable with $\Delta(T, S) \leq \nu$. Hence, we have $\mathbb{E}[\,T^\delta|_{\mathtt{timeout}\mapsto 0}] \leq \mathsf{poly}$. Consequently $\mathbb{E}[T^\delta \mid \neg\mathtt{timeout}] \leq \frac{1}{(1-\delta)}\mathsf{poly} \leq \mathsf{poly}_1$. Let $S = T^\nu$ be the respective ν-quantile of T, note that $S \leq \mathbb{E}[\widetilde{T}]$, and let $\mathcal{G}$ be an event associated with the quantile, (which exists, perhaps after extension of Ω) that is, $S = T|_{\mathcal{G}^\complement \mapsto \mathtt{timeout}}$. Then $\mathbb{E}[T \mid \mathcal{G}] \leq \frac{1}{1-\nu}\mathsf{poly}_1$, and 3 follows.

[33] If neither runtime is efficient, we are in a setting where the truncation argument does not work. Indeed, strings can be encoded as numbers, hence runtimes. Thus, this is indistinguishability of general distributions.

The converse is trivial: If $\mathbb{E}[T \mid \mathcal{G}] \leq \mathsf{poly}$ for an event $\mathcal{G}$ of overwhelming probability $1 - \mathsf{negl}$, then $\widetilde{T} = T|_{\mathcal{G} \mapsto 0}$ is evidently EPT and has statistical distance at most negl. This finishes the equivalence of items 1, 2 and 3.

To see the tail bound, note that for $T \in \mathcal{CEPT}$ there is a "good" runtime $\widetilde{T} \in \mathcal{EPT}$ with $\Delta(T, \widetilde{T}) \leq \mathsf{negl}$. Thus, the tail bound follows immediately from Markov's bound (Lemma C.3.2) applied to $\widetilde{T}$ and statistical distance of negl. That PPT distinguisher suffice and $\mathcal{CEPT}$ is closed was already shown in 5.3.7, but follows easily from the tail bound.

Finally, for induced runtimes, Lemma 5.3.8 demonstrates the equivalence of triple-oracle and standard distinguishing. □

As noted before, non-uniform advice can replace sampling access. For non-uniform distinguishers, triple-oracle and standard indistinguishability coincide. In fact, all the above results follow almost trivially by using the optimal decision table of a distinguisher for $T^{\leq N}$ and $S^{\leq N}$ as advice.

Applications require a further corollary which, though unmotivated, best fits here.

Corollary 5.3.10. *Let* A, B *be two algorithms which output a number in* $\mathbb{N}_0$. *Let* A, B *denote the output distribution and let* $T = \mathsf{time}_{\mathsf{A}}(\mathsf{A})$, $S = \mathsf{time}_{\mathsf{B}}(\mathsf{B})$. *Suppose* $T \overset{d}{\leq}_L q \cdot A$ *and* $S \overset{d}{\leq}_L q \cdot B$ *and let* $L = L(\lambda)$ *and* $q = q(\lambda)$ *polynomial in* λ. *Suppose furthermore* $A \in \mathcal{CEPT}$ *(and hence* $T \in \mathcal{CEPT}$*).*

If $A \overset{c}{\approx} B$ *then* $S \in \mathcal{CEPT}$. *In particular,* $A \in \mathcal{CEPT} \iff B \in \mathcal{CEPT}$ *and statistical and computational (standard and triple-oracle) indistinguishability coincide. The claims generalize to oracle algorithms w.r.t.* $T = \mathsf{time}_{\mathsf{A}}(\mathsf{A}^{\mathcal{O}_A})$, $S = \mathsf{time}_{\mathsf{B}}(\mathsf{B}^{\mathcal{O}_B})$.

The corollary says that, if we measure (and output) a statistic which bounds the runtime (up to polynomial slack), then indistinguishability of that statistic implies preservation of efficiency. This is a core step for the hybrid argument. We stress that the claim is non-trivial, as equivalence of standard and triple-oracle indistinguishability was only proven for *induced* runtimes. Nevertheless, after "rescaling" the tail bound from N to $N \cdot q \cdot L$, the argument is quite analogous to Lemma 5.3.8.

Proof sketch. By assumption, $A \in \mathcal{CEPT}$, and therefore $T \in \mathcal{CEPT}$ (by Lemma C.3.7). By the tail bound for CEPT (see Corollary 5.3.9), for any polynomial $\delta = 1/\mathsf{poly}$, there exists a polynomial N with $\Pr[A > N] < \delta$. Usually, we would choose such an N for suitable small δ and truncate A and B at N, and argue with a standard cutoff argument. However, we cannot truncate A (resp. B) w.r.t. A (resp. B), since these are not the runtimes, and the standard cutoff argument does not apply. Nevertheless, we have a relation between T (resp. S) and A (resp. B) which we can use. So instead, we truncate at K, where $\Pr[A > K] < \frac{\delta}{qL}$. Then

$$\overline{\mathsf{CDF}}_T(K) \leq Lq \cdot \overline{\mathsf{CDF}}_A(K) \leq Lq \frac{\delta}{qL} = \delta.$$

Note that $\frac{\delta}{qL}$ is again polynomial. Thus, we can approximate the distribution of A up to precision δ by using $\mathsf{A}^{\leq K}$.

The first inequality also works with S and B, but as in the standard cutoff argument, we do not know if $\overline{\mathsf{CDF}}_B(K) \leq \frac{\delta}{qL}$. Suppose $\Delta(A, B) > \varepsilon = 1/\mathsf{poly}$ infinitely often. Take $\delta = \varepsilon/4$ and fix the respective cutoff K from above. Then $|\Pr[B > K] - \Pr[A > K]| \leq \varepsilon/4$ or this statistic yields a distinguisher (but we assumed $A \overset{c}{\approx} B$). Hence, $\Delta(A, A') \leq \varepsilon/4$, and $\Delta(B, B') \leq \varepsilon/4$. Thus, we can sample approximations A'

(resp. B') of A (resp. B) via $\mathsf{A}^{\leq K}$ (resp. $\mathsf{B}^{\leq K}$) which are $\varepsilon/4$ statistically close. With this, we can retrace the steps of the CEPT characterization, in particular Lemma 5.3.8, to prove equivalence of triple-oracle and standard indistinguishability. (That is, we approximate the distribution of A', B', and compute from this a decision table for the challenge sample. With non-uniform advice, we can again skip this process, and just assume the advice contains the optimal decisions.)

□

5.3.3. From CEPT to EPT

The characterization of CEPT ensures that, conditioning on "good" events yields a strict EPT algorithm. For interacting parties, this is not yet very useful, because it "entangles" their probability spaces.

Example 5.3.11. Let $\langle \mathsf{P}, \mathsf{V}\rangle$ be an interactive protocol. Suppose P sends a random message $r \in R$. Suppose V picks a random number $s \in R$, and if $r = s$, it loops forever. Otherwise the protocol finishes. Now, the bad event is $\{(r, r) \mid r \in R\}$ (or some superset).

This "entanglement" of probability spaces prevents one core separation, namely the random coins of honest and adversarial parties. Fortunately, they can be "disentangled" as far as possible. Namely, only the (distribution of) *messages* of (honest) parties are of relevance, but no internal coin tosses. This essentially follows from the fact, that the interacting systems have "independent" randomness spaces, and the interaction is mediated solely by messages between the systems.

Lemma 5.3.12 (Timeout oracles). *Let* A *be an interactive algorithm and* $\mathcal{O}$ *be a (probablistic) timeful oracle. Suppose* $\mathrm{time}_{\mathcal{O}}(\langle \mathsf{A}, \mathcal{O}\rangle)$ *is CEPT with virtual runtime* (t, ε). *Then there exists an oracle* $\mathcal{O}'$, *modelled as a* timeful *oracle, such that:* $\mathcal{O}$ *and* $\mathcal{O}'$ *behave identically except when* $\mathcal{O}'$ *sends* `timeout` *(and halts) to signal bad executions. If* A *aborts upon receiving* `timeout`, *then*[34] $\mathrm{time}_{\mathcal{O}}(\langle \mathsf{A}, \mathcal{O}'\rangle)$ *is EPT with expected runtime* $t + \mathcal{O}(1)$ *(with small hidden constant).*[35] *The probability for a* `timeout` *message in* $\langle \mathsf{A}, \mathcal{O}'\rangle$ *is* ε.[36]

We stress that $\mathcal{O}'$ is a *timeful* oracle. While the construction shows that $\mathcal{O}'$ is computable from timed bb-rw access to $\mathcal{O}$, it is generally far from efficiently computable. The usage of Lemma 5.3.12 is roughly as follows: Replace $\mathcal{O}$ with the timeful $\mathcal{O}'$. Now, the runtime problems are is easier to analyse, since we have guaranteed EPT runtime. In the analysis, track the effects on runtime and `timeout` messages of $\mathcal{O}'$. Finally, replace $\mathcal{O}'$ with $\mathcal{O}$ again, noting that only if `timeout` occurs, there is a difference. Of course, such arguments can be made directly, without introducing $\mathcal{O}'$ at all. However, the explicit modification simplifies the presentation.

The construction of $\mathcal{O}'$ is straightforward, one defines $\mathcal{O}'$ by a runtime truncation at N, i.e. $\mathcal{O}'$ acts exactly as $\mathcal{O}$ until the total elapsed time exceeds N. Then, $\mathcal{O}'$ aborts with `timeout`. Exact ν-quantile cutoffs are achieved by extension of $\Omega_{\mathcal{O}}$, as usual.

[34] More formally, one should *lift* A to an algorithm which aborts upon receiving `timeout`, since `timeout` is a special symbol which A cannot receive/interpret.

[35] The constant $\mathcal{O}(1)$ merely accounts for $\mathcal{O}'$ and outputting `timeout`.

[36] The probability space may be enlarged to achieve an exact cutoff, see Section 5.2.5.

Proof of Lemma 5.3.12. A runtime truncation of $\mathcal{O}$ at N is defined in the obvious way, i.e. $\mathcal{O}^{\leq N}$ returns `timeout` if, after an invocation, the purported elapsed runtime exceeds N. An exact ν-quantile cutoff is constructed as usual, i.e. let N be the minimal such that

$$\nu' := \Pr[\langle \mathsf{A}, \mathcal{O}^{\leq N}\rangle \text{ has } \texttt{timeout}] \leq \nu.$$

If this is an equality, let $\mathcal{O}^{\nu}$ be defined as $\mathcal{O}^{\leq N}$. Else, extend $\Omega_{\mathcal{O}}$ via $b \sim \mathrm{Ber}(\nu - \nu')$, so that there is an exact cutoff if one truncates at time t for $t > N$ and for $t = N$ if additionally $b = 1$.

Let $T_{\mathcal{O}} = \mathsf{time}_{\mathcal{O}}(\langle \mathsf{A}, \mathcal{O}\rangle)$. Then

$$T_{\mathcal{O}}^{\nu} = \mathsf{time}_{\mathcal{O}}(\langle \mathsf{A}, \mathcal{O}^{\nu}\rangle),$$

assuming the execution of $\langle \mathsf{A}, \mathcal{O}^{\nu}\rangle$ stops with `timeout` (and the purported runtime is $N = N(\nu)$.) In other words, truncating the runtime distributions and truncating the oracle have the "same" effect.

Our timeout oracle $\mathcal{O}'$ is defined as the ν-quantile truncated oracle, except that $\mathcal{O}'$ additionally pays a small constant time overhead for sending `timeout`. (Recall that due to consistency reasons, sending messages sets lower bounds for purported runtime for timeful oracles.) Note that $\Pr[\langle \mathsf{A}, \mathcal{O}^{\leq N}\rangle \text{ has } \texttt{timeout}] = \nu$ by construction. Moreover

$$\mathsf{time}_{\mathcal{O}}(\langle \mathsf{A}, \mathcal{O}'\rangle) \leq \mathsf{time}_{\mathcal{O}}(\langle \mathsf{A}, \mathcal{O}^{\nu}\rangle) + \mathcal{O}(1),$$

hence the claims follow (as in Corollary 5.3.9). □

In our setting, we usually deal with "multi-oracle" adversaries. For example, zero-knowledge needs input generation $\mathcal{I}$ and a malicious verifier V^* (and a distinguisher $\mathcal{D}$ which of lesser concern). Clearly, we can view $\mathcal{I}$ and V^* as a single oracle (or party), by merging everything except the prover P into one entity. The new entity first runs $\mathcal{I}$ to produce inputs, and then continues as V^*. For completeness, this is discussed more explicitly in Appendix C.5.2.2

5.4. Towards Applications

In this section, we gather the basic tools to deal with a posteriori efficiency. While we focus on CEPT, it will again be evident that our techniques work for "algebra-tailed" runtime classes, and the results generalize to any such class (where the algebra for negligible functions coincides with the algebra for runtime), see Appendix C.4 for the definitions.

5.4.1. Standard Reductions and Truncation Techniques

In this section, we give some semi-abstract reduction and truncation techniques, which are the workhorse for dealing with designated CEPT adversaries.

Lemma 5.4.1 (Reduction to a priori runtime). *Let $\mathcal{O}_0$ and $\mathcal{O}_1$ be two oracles. Suppose $\mathcal{D}$ is a distinguisher with advantage $\varepsilon := \mathrm{Adv}^{dist}_{\mathcal{D},\mathcal{O}_0,\mathcal{O}_1}$. Let $T_0 = \mathsf{time}_{\mathcal{D}}(\mathcal{D}^{\mathcal{O}_0})$ and let $\nu_0 \in [0,1]$ be some negligible function. Suppose there is a tail bound t_0 for T_0 with $\Pr[T_0 > t_0] \leq \frac{1}{4}\varepsilon + \nu$. Then there is a (standard) distinguisher $\mathcal{A}$ with runtime strictly bounded by $t = t_0$ (up to emulation overhead), and advantage at least $\frac{\varepsilon}{4} - \nu_0$ infinitely often. More concretely, $\mathcal{A}$ is a runtime truncation of $\mathcal{D}$ after t steps with tiny overhead.*

The only reason for stating Lemma 5.4.1 in the asymptotic setting is convenience. We also note that instead of runtime, any "runtime-like" statistic, e.g. the number of oracle queries, can be used.

Proof sketch. Let $\mathcal{A}$ run $b \leftarrow \mathcal{D}^{\leq t_0}$ and output b, except if $b = \texttt{timeout}$, where $\mathcal{A}$ returns a random bit instead. The outputs of $\mathcal{D}^{\mathbb{O}_0}$ and $\mathcal{A}^{\mathbb{O}_0}$ have statistical distance at most $\frac{1}{4}\varepsilon + \nu_0$ by assumption on the tail bound t_0.

Suppose the output of $\mathcal{A}^{\mathbb{O}_1}$ has statistical distance δ of $\mathcal{D}^{\mathbb{O}_1}$. If $\delta > \frac{2\varepsilon}{4}$, then necessarily, the probability that $\mathcal{A}^{\mathbb{O}_1}$ exceeds t_0 steps is greater than $\frac{2\varepsilon}{4}$. Thus, this runtime statistic can be used as a distinguishing property, with advantage at least $\frac{\varepsilon}{4} - \nu_0$ (infinitely often). (The distinguisher $\mathcal{A}'$ obtained from this returns 1 on $\texttt{timeout}$ and 0 otherwise.)

Now suppose $\delta \leq \frac{2\varepsilon}{4}$. Then the advantage of $\mathcal{A}$ is at least $\frac{\varepsilon}{4} - \nu_0$ (by statistical distance of the outputs). The promised runtime bounds for $\mathcal{A}$ and $\mathcal{A}'$ follow immediately. □

Plugging in the tail bounds for CEPT, we get the following.

Corollary 5.4.2 (Standard reduction to PPT). *Let $\mathbb{O}_0$ and $\mathbb{O}_1$ be two oracles. Suppose $\mathcal{D}$ a distinguisher with advantage* $\mathrm{Adv}^{dist}_{\mathcal{D},\mathbb{O}_0,\mathbb{O}_1}$ *at least $\varepsilon := \frac{1}{\mathsf{poly}}$ infinitely often, and* $\mathsf{time}_{\mathcal{D}}(\mathcal{D}^{\mathbb{O}_0}) \in \mathcal{CEPT}$. *Then there is an* a priori PPT *(standard) distinguisher $\mathcal{A}$ with advantage at least $\frac{\varepsilon}{4} - \mathsf{negl}$ infinitely often.*

Note that $\mathcal{D}$ need only be efficient for $\mathbb{O}_0$, and that the constructed $\mathcal{A}$ has roughly the same runtime distribution as $\mathcal{D}$.

Remark 5.4.3 (Standard cutoff argument). The strategy in the proof of Lemma 5.4.1 and Corollary 5.4.2 is the *standard cutoff argument.* It works with minor variations in many situations.

Notation 5.4.4. We often sloppily write $\overset{c}{\approx}$ instead of $\overset{c}{\approx}_{\mathcal{T}}$ when specifying indistinguishability. Corollary 5.4.2 justifies this (for the runtime classes of interest).

5.4.2. Relative Efficiency

By considering a posteriori runtime and designated adversaries, we lack a notion of "absolute" efficiency of an algorithm (or timeful system). Instead, we rely on a relative notion of efficiency, which is a definitional cornerstone in our setting.

Definition 5.4.5 (Weak relative efficiency). Let A and B be two (interactive) algorithms (or timeful systems) with identical interfaces. We say that B is **weakly** $(\mathcal{T},\mathcal{S})$**-efficient relative to** A *w.r.t. (implicit) runtime classes* $\mathcal{T}$, $\mathcal{S}$, if for all distinguishing environments $\mathcal{E}$ (which yield closed systems $\langle\mathcal{E},\mathsf{A}\rangle$, $\langle\mathcal{E},\mathsf{B}\rangle$)

$$\mathsf{time}_{\mathcal{E}+\mathsf{A}}(\langle\mathcal{E},\mathsf{A}\rangle) \in \mathcal{T} \implies \mathsf{time}_{\mathcal{E}+\mathsf{A}}(\langle\mathcal{E},\mathsf{B}\rangle) \in \mathcal{S}$$

We say B is **weakly efficient relative to** A *w.r.t. an (implicit) runtime class* $\mathcal{T}$, if it is weakly $(\mathcal{T},\mathcal{T})$-efficient relative to A.

Efficiency relative to a "base" algorithm is the notion of efficiency we need in security definitions and reductions. Indeed, if an adversary is not efficient in the real protocol, the simulator (or reduction) need not be efficient either. However, whenever the adversary is efficient, so should the simulation (or reduction) be.[37] A stronger, unconditional form of relative efficiency is the following. In the following, the runtime classes $\mathcal{T}$ and $\mathcal{S}$ are from $\{\mathcal{PPT}, \mathcal{EPT}, \mathcal{CPPT}, \mathcal{CEPT}\}$, and they decide whether strict or expected time is measured. This allows us to specify cases $\mathcal{T} = \mathcal{S} = \mathcal{PPT}$, $(\mathcal{T}, \mathcal{S}) = (\mathcal{PPT}, \mathcal{EPT})$, and $\mathcal{T} = \mathcal{S} = \mathcal{EPT}$, as well as variations with virtuality succinctly.

Definition 5.4.6 (Tight relative efficiency). Let A, B be as in Definition 5.4.5. We say that B is **$(\mathcal{T}, \mathcal{S})$-efficient relative** to A with **runtime tightness** $(\mathsf{poly}_{\mathsf{time}}, \mathsf{poly}_{\mathsf{virt}})$, if: For all *timeful* environments $\mathcal{E}$, if $\mathsf{time}_{\mathsf{A}}(\langle \mathcal{E}, \mathsf{A}\rangle)$ is virtually strict/expected (t_0, ε_0)-time, then $\mathsf{time}_{\mathsf{B}}(\langle \mathcal{E}, \mathsf{B}\rangle)$ is virtually strict/expected (t_1, ε_1)-time, with $t_1(\lambda) \leq \mathsf{poly}_{\mathsf{time}}(\lambda) t_0(\lambda)$ with $\varepsilon_1(\lambda) \leq \mathsf{poly}_{\mathsf{virt}}(\lambda)\varepsilon_0(\lambda)$ (for all λ).

5.4.3. Hybrid Lemma

The formulation and proof of the hybrid lemma is more involved than for a priori definitions of runtime. To state it, we require relative efficiency. To prove it, we use the random embedding trick from [HUM13] to allow us to get a measure of runtime, which is closely related to the runtime of the full hybrid, even though the time spent in the challenge oracle is inaccessible to a reduction.

Lemma 5.4.7 (Hybrid-Lemma for CEPT). *Suppose that $\mathcal{O}_1$ is weakly efficient relative to $\mathcal{O}_0$ and that $\mathcal{O}_0 \overset{c}{\approx} \mathcal{O}_1$. Suppose that $\mathcal{D}$ is an algorithm with oracle-access to* $\mathsf{rep}(\mathcal{O}_b)$*, and* $\mathsf{time}_{\mathcal{D}+\mathsf{rep}(\mathcal{O}_0)}(\mathcal{D}^{\mathsf{rep}(\mathcal{O}_0)}) \in \mathcal{CEPT}$. *Then* $\mathsf{time}_{\mathcal{D}+\mathsf{rep}(\mathcal{O}_1)}(\mathcal{D}^{\mathsf{rep}(\mathcal{O}_1)}) \in \mathcal{CEPT}$ *and the distinguishing advantage is*

$$\mathrm{Adv}^{dist}_{\mathcal{D},\mathcal{O}_0,\mathcal{O}_1} = |\Pr[\mathcal{D}^{\mathsf{rep}(\mathcal{O}_0)} = 1] - \Pr[\mathcal{D}^{\mathsf{rep}(\mathcal{O}_1)} = 1]| \leq \mathsf{negl}.$$

In other words, $\mathsf{rep}(\mathcal{O}_1)$ *is weakly efficient relative to* $\mathsf{rep}(\mathcal{O}_0)$*, and* $\mathsf{rep}(\mathcal{O}_0) \overset{c}{\approx} \mathsf{rep}(\mathcal{O}_1)$.

For a detailed sketch of the proof and the intuition, we refer back to Section 5.1.4.4.

Proof. We split the proof into several steps. We first show, that proving the claim for q-fold repeated access, for arbitrary but fixed polynomial $q(\lambda)$, is enough.

Claim 5.4.8. *Suppose the hybrid lemma holds for* $\mathsf{rep}_q(\mathcal{O}_0)$ *and* $\mathsf{rep}_q(\mathcal{O}_1)$ *for any polynomial q. That is, for any distinguisher $\mathcal{D}$ with* $\mathsf{time}_{\mathcal{D}+\mathsf{rep}_q(\mathcal{O}_0)}(\mathcal{D}^{\mathsf{rep}_q(\mathcal{O}_0)}) \in \mathcal{CEPT}$, *we have that* $\mathsf{time}_{\mathcal{D}+\mathsf{rep}_q(\mathcal{O}_1)}(\mathcal{D}^{\mathsf{rep}_q(\mathcal{O}_1)}) \in \mathcal{CEPT}$ *and advantage* $\mathrm{Adv}^{dist}_{\mathcal{D},\mathcal{O}_0,\mathcal{O}_1} = |\Pr[\mathcal{D}^{\mathsf{rep}_q(\mathcal{O}_0)} = 1] - \Pr[\mathcal{D}^{\mathsf{rep}_q(\mathcal{O}_1)} = 1]| \leq \mathsf{negl}$. *Then the hybrid lemma holds.*

Proof of Claim 5.4.8. Suppose $\mathcal{D}$ is a distinguisher with $\mathsf{time}_{\mathcal{D}+\mathsf{rep}(\mathcal{O}_0)}(\mathcal{D}^{\mathsf{rep}(\mathcal{O}_0)}) \in \mathcal{CEPT}$ and non-negligible advantage. Let the advantage exceed $\varepsilon = 1/\mathsf{poly}$ infinitely often. The number of Q_0 of $\mathcal{O}_0$-instances generated in $\mathcal{D}^{\mathsf{rep}(\mathcal{O}_0)}$ is certainly CEPT. Let Q_1 denote the number of $\mathcal{O}_1$-instances generated in $\mathcal{D}^{\mathsf{rep}(\mathcal{O}_1)}$. Treating these statistics as "runtime", the standard truncation argument (Corollary 5.4.2), ensures that there is a PPT distinguisher $\mathcal{A}$ which makes a strictly polynomial number q of queries and

[37] Strictly speaking, a simulator depends on the adversary, and Definition 5.4.5 should be applied "pointwise", i.e. for every adversary, the simulator should be efficient relative to the simulator. For completeness, we give a generalized definition in Appendix C.5.3.3.

has advantage at least $\varepsilon/4 - \mathsf{negl}$ infinitely often. Clearly, $\mathcal{A}^{\mathsf{rep}(\mathcal{O}_0)}$ has runtime bounded by $\mathcal{D}^{\mathsf{rep}(\mathcal{O}_0)}$ (up to emulation overhead), hence remains efficient. Consequently, we have reduced to the setting where at most q queries are made for some polynomial q which depends on $\mathcal{D}$. □

From now on, we assume that $\mathcal{D}$ generates at most q oracle instances, for some polynomial q. We proceed in game hops, starting with $\mathcal{D}^{\mathsf{rep}(\mathcal{O}_0)}$ and finishing with $\mathcal{D}^{\mathsf{rep}(\mathcal{O}_1)}$ For each hop, we have to ensure indistinguishable output bits and preservation of efficiency.

- **Game** G_0 is simply the execution of $\mathcal{D}^{\mathsf{rep}(\mathcal{O}_0)}$.
- In **Game** G_1 we pick a random permutation $\pi\colon \{1,\dots,q\} \to \{1,\dots,q\}$ and reroute the access to the oracles: If $\mathcal{D}$ queries for the i-th oracle, it is routed to the $\pi(i)$-th oracle. More precisely, in G_0 the adversary $\mathcal{D}$ has access to $\vec{\mathcal{O}} = (\mathcal{O}_0^1,\dots,\mathcal{O}_0^q)$, whereas in G_1 it has access to a random permutation $\vec{\mathcal{O}}^{\pi} = (\mathcal{O}_0^{\pi(1)},\dots,\mathcal{O}_0^{\pi(q)})$. Clearly, G_0 and G_1 are perfectly indistinguishable and have almost identical runtime (up to bookkeeping). The key change is, that all $\mathcal{O}^i$ now have identical runtime distributions.
- In **Game** G_2, we replace the $\mathcal{O}_0^1$ with $\mathcal{O}_1^1$, that is we consider $\vec{\mathcal{O}} = (\mathcal{O}_1^1, \mathcal{O}_0^2, \dots, \mathcal{O}_0^q)$. Indistinguishability (of outputs) of G_2 follows directly from the standard reduction, whereas efficiency of G_2 follows from $\mathcal{O}_1$ being weakly efficient relative to $\mathcal{O}_0$. Note that the runtime of G_2 may differ significantly from that of G_1.
- In **Game** G_3, we have $\vec{\mathcal{O}} = (\mathcal{O}_1^1,\dots,\mathcal{O}_1^{q-1},\mathcal{O}_0^q)$. That is, all but one oracle instance is of $\mathcal{O}_1$-type. Proving that G_3 is CEPT is the key point in this argument. Indistinguishability of G_2 and G_3 follows easily. We postpone the proof to Claim 5.4.9, and finish up first.
- In **Game** G_4, we use $\vec{\mathcal{O}} = (\mathcal{O}_1^1,\dots,\mathcal{O}_1^{q-1},\mathcal{O}_1^q)$, that is, we switched completely to $\mathcal{O}_1$ for every instance. Efficiency and output indistinguishability follow as from G_1 to G_2.
- In **Game** G_5, we remove the random permutation π. Thus, G_5 is $\mathcal{D}^{\mathsf{rep}(\mathcal{O}_1)}$, as claimed. □

Claim 5.4.9. *If* G_2 *is CEPT, so is* G_3*. Moreover, their outputs are indistinguishable.*

We will prove Claim 5.4.9 by establishing a relatively precise grasp on the runtime.

Proof. Recall that we have to switch the oracle setup from $(\mathcal{O}_1^1, \mathcal{O}_0^2, \dots, \mathcal{O}_0^q)$ to $(\mathcal{O}_1^1,\dots,\mathcal{O}_1^{q-1},\mathcal{O}_0^q)$. The core difficulty is the efficiency in the latter case. Following the trick of [HUM13], we randomized the oracle order for $\mathcal{D}$ using a random permutation π. This spreads the runtime of $\mathcal{O}_1^1$ and $\mathcal{O}_0^q$ evenly over all possible positions, and this property is at the heart of the reduction.

Let H_ℓ denote the game with oracle setup $\vec{\mathcal{O}} = (\mathcal{O}_1^1,\dots,\mathcal{O}_1^\ell,\mathcal{O}_0^{\ell+1},\dots,\mathcal{O}_0^q)$.[38] By construction, H_1 equals G_2 and H_{q-1} equals G_3. Thus, it suffices to prove indistinguishability of H_1 and H_{q-1}.

Clearly, hybrids H_ℓ and $\mathsf{H}_{\ell+1}$ (for $\ell = 1,\dots,q-2$) are related by a direct reduction to $\mathcal{O}_0 \overset{c}{\approx} \mathcal{O}_1$. The hybrid reduction R embeds the challenge oracle $\mathcal{O}_b^*$ in position $\ell+1$, picking $\ell \leftarrow \{1,\dots,q-2\}$ uniformly. Denote by R_ℓ the reduction with fixed choice ℓ. By construction, $R_\ell^{\mathcal{O}_b^*} = \mathsf{H}_{\ell+b}$. Note that $\mathcal{D}$ has randomly permuted access, so the challenge oracle is embedded uniformly from $\mathcal{D}$'s view.

[38] Note that all hybrids have $\mathcal{O}_1^1$ and $\mathcal{O}_0^q$ fixed. Thus there $q-1$ hybrids and $q-2$ hybrid transitions.

One main complication is, that $R_\ell^{\mathcal{O}^*}$ cannot keep runtime statistics of $\mathcal{O}^*$. Yet, we need enough control over the runtime to guarantee efficiency of R and H_{q-1}. By randomizing the order of the oracle instances (from the view of $\mathcal{D}$), we can exploit the strong symmetry of the local runtimes of instances. Let us now take a close look at these runtimes. To keep notation in check, we fix a hybrid H_ℓ, and notationally suppress the dependency of most variables on ℓ.

- Let $T_\ell = \mathsf{time}(\mathsf{H}_\ell)$ be the total runtime of H_ℓ (as a random variable). Recall that it is understood that H_ℓ emulates all oracles $\mathcal{O}^1, \dots, \mathcal{O}^q$.
- Let $T^{\mathcal{O},j}$ denote the time H_ℓ spends in $\mathcal{O}^j$.
- Let $T^{\mathcal{D}}$ denote the time H_ℓ spends outside $\mathcal{O}^j$ (mostly emulating $\mathcal{D}$).
- We have $T_\ell = T^{\mathcal{D}} + \sum_{j=1}^q T^{\mathcal{O},j}$ as random variables by definition.

By the symmetry introduced by the random permutation π, the distributions of the $T^{\mathcal{O},j}$ for the same type of oracle coincide. That is, $T^{\mathcal{O},i} \overset{d}{\equiv} T^{\mathcal{O},j}$ for all $(i, j) \in \{1, \dots, \ell\}$ for $\mathcal{O}_1$-type instances, and likewise with $(i, j) \in \{\ell + 1, \dots, q\}$ for all $\mathcal{O}_0$-type instances.

Claim 5.4.10. *Let $S_\ell = T^{\mathcal{D}} + T^{\mathcal{O},1} + T^{\mathcal{O},q}$. Then we have*

$$\overline{\mathsf{CDF}}_{T_\ell}(\,\cdot\,) \le (q+1)\overline{\mathsf{CDF}}_{(q+1)S_\ell}(\,\cdot\,) \quad \textit{that is} \quad T_\ell \overset{d}{\le}_{(q+1)} (q+1) \cdot S_\ell. \tag{5.4.1}$$

Proof. Using the definition and symmetries, we argue that

$$\begin{aligned}
\overline{\mathsf{CDF}}_{T_\ell}(t) &= \Pr[T^{\mathcal{D}} + \sum_{i=1}^{\ell} T^{\mathcal{O},i} + \sum_{j=\ell+1}^{q} T^{\mathcal{O},j} > t] \\
&\le \Pr[T^{\mathcal{D}} > \frac{1}{q+1}t] + \Pr[\sum_{i=1}^{\ell} T^{\mathcal{O},i} > \frac{\ell}{q+1}t] + \Pr[\sum_{j=\ell+1}^{q} T^{\mathcal{O},j} > \frac{q-\ell}{q+1}t] \\
&\le \Pr[T^{\mathcal{D}} > \frac{1}{q+1}t] + \sum_{i=1}^{\ell} \Pr[T^{\mathcal{O},i} > \frac{1}{q+1}t] + \sum_{j=\ell+1}^{q} \Pr[T^{\mathcal{O},j} > \frac{1}{q+1}t] \\
&= \Pr[(q+1) \cdot T^{\mathcal{D}} > t] + \ell \cdot \Pr[(q+1) \cdot T^{\mathcal{O},1} > t] + (q-\ell) \cdot \Pr[(q+1) \cdot T^{\mathcal{O},q} > t] \\
&\le (q+1) \cdot \Pr[(q+1) \cdot S_\ell > t].
\end{aligned}$$

The first two inequalities use that for any sum $\sum_{i=1}^n \lambda_i X_i > x$ with $\lambda_i \ge 0$ and $\sum_{i=1}^n \lambda_i = 1$, there exists some i such that $X_i > \lambda_i x$. The next (in)equalities follow from symmetries and simplifications. The final inequality holds, because by construction, S_ℓ dominates $T^{\mathcal{D}}$, $T^{\mathcal{O},1}$ and $T^{\mathcal{O},q}$. Also, we bound 1, ℓ and $q - \ell$ by $q + 1$. Thus the claim follows. □

Note that S_ℓ can be computed, even in a reduction between hybrids, since $\mathcal{O}^1$ and $\mathcal{O}^q$ are fixed and there is never a challenge-embedding there. This is crucial. Using Eq. (5.4.1), the generalized CEPT characterization, Corollary 5.3.10, is applicable. Therefore, it suffices to prove that $S_1 \overset{c}{\approx} S_{q-1}$, and Corollary 5.3.10 ensures that T_{q-1} is CEPT if T_1 is CEPT This is our next step. (In a sense, we have now reduced efficiency to indistinguishability.)

Claim 5.4.11. *For S_ℓ defined as above, we have $S_1 \overset{c}{\approx} S_{q-1}$.*

To prove $S_1 \overset{c}{\approx} S_{q-1}$, we modify the hybrids H_ℓ to also output this quantity. That is, the hybrid H_ℓ outputs the runtime S_ℓ and output of $\mathcal{D}$, obtained by emulating $\mathcal{D}$ given access to $\vec{\mathcal{O}}^\pi$ with $\vec{\mathcal{O}} = (\mathcal{O}_1^1, \dots, \mathcal{O}_1^\ell, \mathcal{O}_0^{\ell+1}, \dots, \mathcal{O}_0^q)$ for $\ell = 1, \dots, q-1$. For now, we focus solely on the output S_ℓ.

Recall that R denotes the hybrid reduction, which embeds its challenge-oracle $\mathcal{O}^*$ into $\mathcal{O}^{i^*}$ for $i^* = i+1$, where $i \leftarrow \{1, \dots q-2\}$), and simulates the remaining oracles. Recall, that R_i denotes reduction R with fixed choice i. Hence, $R_\ell^{\mathcal{O}_b^*}$ simulates $\mathsf{H}_{\ell+b}$.

If $R^{\mathcal{O}_0}$ were efficient, this would almost finish the proof. However, we do not yet know whether $R^{\mathcal{O}_0}$ is efficient. Since we only need to prove S_1 and S_{q-1} indistinguishable, we truncate the hybrids and the reduction. We then need to prove that the truncations are close to the originals, i.e. the probability for `timeout` is (arbitrarily polynomially) small. We define:

- $[\mathsf{H}_\ell]$ is the hybrid which imposes a strict time bound of $t_{\max}$ (to be chosen later) on each oracle emulation (i.e. each $\mathcal{O}^i$) individually as well as the emulation of $\mathcal{D}$. If a bound is exceed, $[\mathsf{H}_\ell]$ aborts with `timeout`. (That is, $[\mathsf{H}_\ell]$ aborts if $T^{\mathcal{D}} > t_{\max}$ or $T^{\mathcal{O},i} > t_{\max}$ for any $i = 1, \dots, q$.)
- $[R]_\ell$ is defined analogously to $[\mathsf{H}_\ell]$. Note that $[R]$ cannot truncate its challenge oracle $\mathcal{O}^*$.
- $[\mathcal{O}]$ (resp. $[\mathcal{D}]$) denotes same cutoff at $t_{\max}$ applied to an oracle (resp. $\mathcal{D}$).

By definition

$$[\mathsf{H}_\ell] = [R]_{\ell-1}^{[\mathcal{O}_1]} = [R]_\ell^{[\mathcal{O}_0]}$$

if the expressions are defined. The technical problem, is that we can only compute $[R]_\ell^{\mathcal{O}^*}$, but not $[R]_\ell^{[\mathcal{O}^*]}$. Yet, the latter is necessary in the usual telescoping sum. To quantify the introduced error, we define:

- $h_\ell := \Pr[[\mathsf{H}_\ell] = \mathtt{timeout}]$, the `timeout` probability of $[\mathsf{H}_\ell]$.
- $r_\ell^b := \Pr[[R]_\ell^{\mathcal{O}_b} = \mathtt{timeout}]$, the `timeout` probability of $[R]_\ell^{\mathcal{O}_b}$. Note that the challenge oracle $\mathcal{O}_b$ cannot time out.

Claim 5.4.12. *For all $\delta = 1/\mathrm{poly}$, there exists a polynomial $t_{\max}$ such that for all $\ell = 1, \dots q-1$*

$$h_\ell \le \delta \quad \textit{and} \quad |h_\ell - r_{\ell-1}^1| \le \delta \quad \textit{and} \quad |h_\ell - r_\ell^0| \le \delta.$$

Similar to before, and as in [HUM13], it is easy to use the symmetry of `timeouts` to show[39]

$$r_{\ell-1}^1 \le h_\ell \le \frac{\ell}{\ell-1} \cdot r_{\ell-1}^1 \tag{5.4.2}$$

$$r_\ell^0 \le h_\ell \le \frac{q-\ell}{q-\ell-1} \cdot r_\ell^0 \tag{5.4.3}$$

for all $\ell = 1, \dots, q-2$. This implies

$$h_\ell \le \frac{\ell}{\ell-1} r_{\ell-1}^1 = \frac{\ell}{\ell-1} r_{\ell-1}^0 + \frac{\ell}{\ell-1}(r_{\ell-1}^1 - r_{\ell-1}^0) \le \frac{\ell}{\ell-1} h_{\ell-1} + \frac{\ell}{\ell-1} \rho_{\ell-1}$$

[39] This is the reason we applied `timeouts` to each oracle individually, instead of to the whole game H_ℓ. The latter may not exhibit this symmetry.

where we let $\rho_i = r_i^1 - r_i^0$. Inductively we find[40]

$$h_\ell \leq \ell \cdot h_1 + \sum_{i=1}^{\ell-1} \frac{\ell}{i} \rho_i \tag{5.4.4}$$

Recall that we can make h_1 arbitrarily polynomially small by picking $t_{\max}$ large enough. Now, we want to prove that h_ℓ is also small for all ℓ. Hence we are looking for a (small) upper bound on $\sum_{i=1}^{\ell-1} \frac{\ell}{i} \rho_i$.

If all ρ_i were positive, we could just guess a good i and use a completely standard hybrid argument, but we do not know this. In [HUM13], non-uniform advice is used, namely the index i^* which maximizes $|\rho_{i^*}|$. It is easy to see that $[R]_{i^*}$ is a distinguisher with advantage $|\rho_{i^*}|$, hence $|\rho_{i^*}|$ is negligible and consequently $|\sum_{i=1}^{\ell-1} \frac{\ell}{i} \rho_i| \leq q \cdot |\rho_{i^*}|$ is also negligible. It is also noted in [HUM13], that one can approximate ℓ^* using $[R]^{\mathbb{O}_b}$ to obtain a uniform distinguisher. In [HUM13], $[R]^{\mathbb{O}_0}$ is efficient by assumption, which makes approximation of i^* straightforward. In our setting, efficiency of $[R]^{\mathbb{O}_0}$ does not hold by assumption. Thus, we need to argue differently.

We approximate a good index i^*, where ρ_{i^*} is large by using the following inequality

$$\rho_{\ell-1}^- := \frac{\ell-1}{\ell} h_\ell - h_{\ell-1} \leq \rho_{\ell-1} = r_{\ell-1}^1 - r_\ell^0 \tag{5.4.5}$$

where we used Eqs. (5.4.2) and (5.4.3). Observe that $\mathrm{Ber}(h_i)$ can be sampled in $\mathrm{time}([\mathsf{H}_i]) \leq (q+1) \cdot t_{\max}$ (up to emulation overhead), since $h_i = \Pr[[\mathsf{H}_i] = \mathtt{timeout}]$. Hence, we can approximate h_i (and hence ρ_i^-) via sampling to arbitrary polynomial precision. By induction, we find

$$h_\ell = \ell \cdot h_1 + \sum_{i=1}^{\ell-1} \frac{\ell}{i} \rho_i^- \tag{5.4.6}$$

which is an equality by definition. Note that if $\max_{i=1}^{q-2} \rho_i^- \leq \nu$ for some negligible ν, we get $h_\ell \leq \ell h_1 + \ell^2 \nu$. This is sufficient for our purposes. We stress that we do not consider absolute values here, as we only need an upper bound for the timeout probability.[41]

We argue by contradiction. Suppose $\max_{i=1}^{q-2} \rho_i^- > 1/\mathsf{poly}$ infinitely often. Then, as noted, we can approximate ρ_i^- up to any polynomial precision, and in particular we can sample a "good" i^* which satisfies $\rho_{i^*} \geq \rho_{i^*}^- \geq \frac{1}{2\mathsf{poly}(\lambda)}$ with overwhelming probability.[42] Thus, this yields a distinguisher for $\mathbb{O}_0$ and $\mathbb{O}_1$, which is efficient for any choice of polynomial cutoff bound $t_{\max}$ and has advantage $\geq \frac{1}{2\mathsf{poly}} - \mathsf{negl}$ infinitely often. (Namely, first approximate i^* and then run $[R]_{i^*}^{\mathbb{O}^*}$. The choice of i^* is "good" with overwhelming probability, so we get an advantage of at least $\frac{1}{2\mathsf{poly}(\lambda)} - \mathsf{negl}$ infinitely often.) This finally proves that

$$\forall t_{\max} = \mathsf{poly}\ \exists \nu = \mathsf{negl}\ \forall \ell = 2, \dots, q-1 \colon \quad h_\ell \leq \ell h_1 + \ell^2 \nu.$$

In particular, for almost all λ, $h_\ell \leq q h_1 + q^2 \nu$. Since H_1 is CEPT, for any poly there is a $t_{\max}$ such that $h_1 \leq 1/\mathsf{poly}$. Let $\delta = 1/\mathsf{poly}$ for some poly. Pick $t_{\max}$ such that $h_1 \leq \frac{\delta}{2q}$. Then $h_\ell \leq \delta/2 + q^2\nu \leq \delta$ for almost all λ. This proves the first part of Claim 5.4.12. From Eqs. (5.4.2) and (5.4.3) we also find

$$|h_\ell - r_\ell^0| \leq \frac{1}{q-\ell-1} h_\ell \quad \text{and} \quad |h_\ell - r_{\ell-1}^1| \leq \frac{1}{\ell-1} h_\ell.$$

[40] In a standard hybrid argument, we would have $\sum_i \rho_i = r_1^0 - r_{\ell-1}^1$ instead of a weighted sum.
[41] One can similarly define $\rho_\ell^+ := h_{\ell+1} - \frac{q-\ell-1}{q-\ell} h_\ell \geq \rho_\ell$ and consider $\min_{i=2}^{q-1} \rho_i^-$. Hence there is a negl such that $|\rho_\ell| \leq \mathsf{negl}$ for all ℓ. We do not need this.
[42] We can use the same approximation of distributions as for CEPT, see also Appendix C.3.4.

This completes the proof of Claim 5.4.12. Claim 5.4.11 now follows from Lemma 5.4.13 below. More concretely, it follows that $\mathsf{H}_1 \overset{c}{\approx} \mathsf{H}_{q-1}$, which implies $S_1 \overset{c}{\approx} S_{q-1}$ (since we augmented the output of H_ℓ by S_ℓ), and, by Corollary 5.3.10, H_{q-1} is CEPT (since H_1 is CEPT). This finishes the proof of Claim 5.4.9.

□

The following lemma uses notation similar to the above, but does not follow the indexing. This simplifies the presentation, as we can go from 0 to q instead of 1 to $q-1$.

Lemma 5.4.13 (Approximable hybrid lemma). *Let $\mathcal{O}_0$, $\mathcal{O}_1$ be oracles, let $\mathsf{H}_0, \dots, \mathsf{H}_q$ be hybrid games (for polynomial q) and let R be an algorithm. We call R a* hybrid reduction *for* H*, if it is of the following form:*

- *R is an oracle algorithm which in the beginning chooses a random integer $i^* \in \{0, \dots, q-1\}$.*
- *Denoting by R_i the algorithm with fixed choice $i^* = i$, we have $R_i^{\mathcal{O}_b} \equiv \mathsf{H}_{i+b}$, where we mean equivalence as systems.*[43]

We say R is a time-approximable hybrid reduction*, if for any choice $\delta = \frac{1}{\mathsf{poly}_\delta}$:*

1. *There exist "truncated" a priori PPT hybrids $[\mathsf{H}_\ell]$ for $i = 0, \dots, q$, which may return* `timeout`*.*
2. *H_ℓ and $[\mathsf{H}_\ell]$ are equal until* `timeout` *and* $\Pr[[\mathsf{H}_\ell] = \texttt{timeout}] \leq \delta$.
3. *There exists a "truncated" a priori PPT reduction algorithm $[R]$ with $\Delta([\mathsf{H}_\ell], [R]_\ell^{\mathcal{O}_0}) \leq \delta$ and $\Delta([\mathsf{H}_{\ell+1}], [R]_\ell^{\mathcal{O}_1}) \leq \delta$ (for almost all λ).*

If R is a time-approximable hybrid reduction for H, then $R_0^{\mathcal{O}_0} \equiv \mathsf{H}_1 \overset{c}{\approx} \mathsf{H}_q \equiv R_{q-1}^{\mathcal{O}_1}$. More precisely, for every a priori PPT distinguisher $\mathcal{D}$ and for every $\delta = \frac{1}{\mathsf{poly}_\delta}$ there exists an adversary $\mathcal{A}$ such that $\mathrm{Adv}^{dist}_{\mathsf{H}_0,\mathsf{H}_q,\mathcal{D}} \leq q \cdot \mathrm{Adv}^{dist}_{\mathcal{O}_0,\mathcal{O}_1,\mathcal{A}} + (2q+2)\delta$ (for almost all λ).

One can replace item 3 with

4. *For any $\delta = \frac{1}{\mathsf{poly}_\delta}$ there exists a "truncated" a priori PPT reduction algorithm $[R]$, such that $\Delta(R_\ell^{\mathcal{O}_0}, [R]_\ell^{\mathcal{O}_0}) \leq \delta$ for all ℓ.*

In this case, we get $\mathrm{Adv}^{dist}_{\mathsf{H}_0,\mathsf{H}_q,\mathcal{D}} \leq q \cdot \mathrm{Adv}^{dist}_{\mathcal{O}_0,\mathcal{O}_1,\mathcal{A}} + (4q+4)\delta$ (for almost all λ).

Note that Claim 5.4.11 establishes items 1, 2 and 4 in the respective setting. Hence, Lemma 5.4.13 is applicable.

One can easily generalize Lemma 5.4.13 beyond runtime truncation. Truncation and `timeout` ("time-approximation") is just the special case of approximation we are most interested in.

Proof. We can assume w.l.o.g. that H_ℓ outputs a bit, since we can integrate a distinguisher (which is w.l.o.g. a priori PPT) into H. Thus, the distinguisher advantage is now simply $\Delta(\mathsf{H}_0, \mathsf{H}_q)$. Consider some $\delta = \frac{1}{\mathsf{poly}_\delta}$ and let $[\mathsf{H}_\ell]$ and $[R]_\ell$ as in the statement. By the triangle inequality and item 2, we get

$$\Delta(\mathsf{H}_0, \mathsf{H}_q) \leq \Delta([\mathsf{H}_0], [\mathsf{H}_q]) + 2\delta.$$

[43] We assume that H_ℓ is a closed system. But this actually unnecessary, if we allow distinguishing environments.

Moreover, we have for almost all λ

$$\begin{aligned}
\Delta([\mathsf{H}_0],[\mathsf{H}_q]) &= \Big|\sum_{\ell=0}^{q-1} \Pr[[\mathsf{H}_\ell]=1] - \Pr[[\mathsf{H}_{\ell+1}]=1]\Big| \\
&= \Big|\sum_{\ell=0}^{q-1} \Pr[[\mathsf{H}_\ell]=1] - \Pr[[R]_\ell^{\mathcal{O}_0}=1] + \Pr[[R]_\ell^{\mathcal{O}_1}=1] \\
&\quad - \Pr[[\mathsf{H}_{\ell+1}]=1] + \Pr[[R]_\ell^{\mathcal{O}_0}=1] - \Pr[[R]_\ell^{\mathcal{O}_1}=1]\Big| \\
&\leq \sum_{\ell=0}^{q-1} |\Pr[[\mathsf{H}_\ell]=1] - \Pr[[R]_\ell^{\mathcal{O}_0}=1]| \\
&\quad + \sum_{\ell=0}^{q-1} |\Pr[[R]_\ell^{\mathcal{O}_1}=1] - \Pr[[\mathsf{H}_{\ell+1}]=1]| \\
&\quad + |\sum_{\ell=0}^{q-1} \Pr[[R]_\ell^{\mathcal{O}_0}=1] - \Pr[[R]_\ell^{\mathcal{O}_1}=1]| \\
&\leq q\delta + q\delta + q\cdot \mathrm{Adv}^{\mathrm{dist}}_{\mathcal{O}_0,\mathcal{O}_1,\mathcal{A}}(\lambda)
\end{aligned}$$

where the inequality follows from item 3 and the "hybrid reduction" adversary $\mathcal{A}$ which runs $d \longleftarrow [R]^{\mathcal{O}^*}$ and if $d \neq \texttt{timeout}$ outputs d, else 1. Taken together, we find for almost all λ

$$\Delta(\mathsf{H}_0,\mathsf{H}_q) \leq q\cdot \mathrm{Adv}^{\mathrm{dist}}_{\mathcal{O}_0,\mathcal{O}_1,\mathcal{A}}(\lambda) + (2q+2)\delta.$$

Since $\mathcal{O}_0 \overset{c}{\approx} \mathcal{O}_1$, $\mathrm{Adv}^{\mathrm{dist}}_{\mathcal{O}_0,\mathcal{O}_1,\mathcal{A}}(\lambda)$ is negligible. Now, let $\varepsilon = 1/\mathsf{poly}$ some prescribed polynomial bound. Since q is polynomial and δ is chosen after q, one can choose $\delta^{-1} \geq (4q+4)\varepsilon^{-1}$, which is still polynomial and ensures that $(2q+2)\delta \leq \frac{1}{2}\varepsilon$. Hence, $\Delta(\mathsf{H}_0,\mathsf{H}_q) = \frac{1}{2}\varepsilon + q\cdot\mathsf{negl} < \varepsilon$ (for almost all λ). Thus, $\Delta(\mathsf{H}_0,\mathsf{H}_q)$ is smaller than any polynomial ε. Consequently, $\Delta(\mathsf{H}_0,\mathsf{H}_q)$ is negligible, and the first part of the claim follows.

For the second part, note that from items 2 and 4 we find

$$\Delta([\mathsf{H}_{\ell+b}],[R]_\ell^{\mathcal{O}_b}) \leq \Delta([\mathsf{H}_{\ell+b}],\mathsf{H}_\ell) + \Delta(\mathsf{H}_{\ell+b},R_\ell^{\mathcal{O}_b}) + \Delta(R_\ell^{\mathcal{O}_b},[R]_\ell^{\mathcal{O}_b}).$$

By assumption $\Delta(\mathsf{H}_{\ell+b},R_\ell^{\mathcal{O}_b}) = 0$. Thus, if $\Delta([\mathsf{H}_\ell],\mathsf{H}_\ell) \leq \delta/2$ and $\Delta(R_\ell^{\mathcal{O}_b},[R]_\ell^{\mathcal{O}_b}) \leq \delta/2$ for all ℓ and $b \in \{0,1\}$, then $\Delta([\mathsf{H}_\ell],[R]_\ell^{\mathcal{O}_b}) \leq \delta$ and hence item 3 is satisfied. The claim now follows by using $\delta' = \delta/2$ as choice of δ for $[\mathsf{H}_\ell]$ and $[R]_\ell$ in items 2 and 4. □

5.5. Application to Zero-Knowledge Arguments

Our flavour of zero-knowledge follows Goldreich's treatment of uniform complexity [Gol93], combined with Feige's designated adversaries [Fei90]. We only define efficient proof systems for NP-languages.

Definition 5.5.1 (Interactive arguments). Let $\mathcal{R}$ be an NP-relation with corresponding language $\mathcal{L}$. An **argument (system) for $\mathcal{L}$** consists of two interactive algorithms (P,V) such that:

Efficiency: There is a polynomial poly so that for all $(\lambda,\mathbb{x},\mathbb{w})$ the runtime $\mathsf{time}_{\mathsf{P}+\mathsf{V}}(\langle \mathsf{P}(\mathbb{x},\mathbb{w}),\mathsf{V}(\mathbb{x})\rangle)$ is bounded by $\mathsf{poly}(\lambda,|\mathbb{x}|)$.

Completeness: $\forall(\mathbb{x},\mathbb{w}) \in \mathcal{R} \colon \mathsf{out}_{\mathsf{V}}\langle \mathsf{P}(\mathbb{x},\mathbb{w}), \mathsf{V}(\mathbb{x})\rangle = 1$.

Definition 5.5.1 essentially assumes "classic" PPT algorithms, but it will be evident that our techniques do not require this. We do not define soundness, but note that it is easily handled via truncation to a PPT adversary. The terms proof and argument systems are often used interchangeably (and we also do this). Strictly speaking, *proof* systems require unconditional soundness and allow unbounded provers. Argument systems allow computational soundness and require efficient provers. All our exemplary proof systems [GMW86; GK96; Lin13; Ros04; KP01; PTV14] have efficient provers, hence are also argument systems.

5.5.1. Zero-Knowledge

Definition 5.5.2. Let $\mathcal{T}, \mathcal{S} \in \{\mathcal{PPT}, \mathcal{CPPT}, \mathcal{EPT}, \mathcal{CEPT}\}$. Let (P,V) be an argument system. A **universal simulator** Sim takes as input $(\mathsf{code}(\mathsf{V}^*), \mathbb{x}, \mathit{aux})$ and simulates V^*'s output. Let $(\mathcal{I}, \mathsf{V}^*, \mathcal{D})$ be an adversary. We define the real and ideal executions as

$$\begin{aligned} \mathsf{Real}_{\mathcal{I},\mathsf{V}^*}(\lambda) &\coloneqq (\mathit{state}, \mathsf{out}_{\mathsf{V}^*}\langle \mathsf{P}(\mathbb{x},\mathbb{w}), \mathsf{V}^*(\mathbb{x},\mathit{aux})\rangle) \\ \text{and} \quad \mathsf{Ideal}_{\mathcal{I},\mathsf{Sim}(\mathsf{code}(\mathsf{V}^*))}(\lambda) &\coloneqq (\mathit{state}, \mathsf{Sim}(\mathsf{code}(\mathsf{V}^*), \mathbb{x}, \mathit{aux})) \end{aligned}$$

where $(\mathbb{x},\mathbb{w},\mathit{aux},\mathit{state}) \leftarrow \mathcal{I}$ and $(\mathbb{x},\mathbb{w}) \in \mathcal{R}$, else Real and Ideal return a failure symbol, say $\bot$. We omit the input $\mathsf{code}(\mathsf{V}^*)$ to Sim, if it is clear from the context. The advantage of $(\mathcal{I}, \mathsf{V}^*, \mathcal{D})$ is

$$\mathrm{Adv}^{\mathrm{zk}}_{\mathcal{I},\mathsf{V}^*,\mathcal{D}}(\lambda) \coloneqq |\Pr[\mathcal{D}(\mathsf{Real}_{\mathcal{I},\mathsf{V}^*}(\lambda)) = 1] - \Pr[\mathcal{D}(\mathsf{Ideal}_{\mathcal{I},\mathsf{Sim}}(\lambda)) = 1]|.$$

A (designated) adversary $(\mathcal{I}, \mathsf{V}^*, \mathcal{D})$ is $\mathcal{T}$-time if $\mathsf{time}_{\mathcal{I}+\mathsf{P}+\mathsf{V}^*+\mathcal{D}}(\mathcal{D}(\mathsf{Real}_{\mathcal{I},\mathsf{V}^*})) \in \mathcal{T}$.

The argument is **(uniform) (auxiliary input) zero-knowledge** against $\mathcal{T}$-time adversaries w.r.t. $\mathcal{S}$-time Sim, if for any $\mathcal{T}$-time adversary $(\mathcal{I}, \mathsf{V}^*, \mathcal{D})$:

- $\mathsf{time}_{\mathcal{I}+\mathsf{Sim}+\mathcal{D}}(\mathcal{D}(\mathsf{Ideal}_{\mathcal{I},\mathsf{Sim}})) \in \mathcal{S}$. The runtime of Sim includes whatever time is spent to emulate V^*. In a (generalized) sense, Sim is weakly $(\mathcal{T},\mathcal{S})$-efficient relative to P, see Definition C.5.12.
- $\mathrm{Adv}^{\mathrm{zk}}_{\mathcal{I},\mathsf{V}^*,\mathcal{D}}(\lambda)$ is negligible

Some more remarks are in order.

Remark 5.5.3. In our setting, *existential* and *universal* simulation are equivalent. The adversary $\mathcal{V}_{\mathrm{univ}}$, which executes *aux* as its code, is universal, see Appendix C.5.4.4.

Remark 5.5.4 (Reductions to PPT). By a standard reduction to PPT, we may w.l.o.g. assume that $\mathcal{D}$ is a priori PPT. Perhaps surprisingly, this is false for $\mathcal{I}$. Intuitively, verifying the *quality* of the output of Sim requires only PPT $\mathcal{D}$ (and $\mathcal{I}$). Verifying the *efficiency*, however, does not. The cause is that $\mathcal{I}$ may generate *expected* poly-size inputs. See also Example C.5.26.

Remark 5.5.5 (Efficiency of the simulation). Definition 5.5.2 only ensures that Sim is *weakly* efficient relative to P, i.e. we have no tightness bounds. Relative efficiency with tightness bounds is an unconditional property, and hence not possible if zero-knowledge holds only computationally.

In the definition, it is possible to replace $\mathsf{time}_{\mathcal{I}+\mathsf{Sim}+\mathcal{D}}(\mathcal{D}(\mathsf{Ideal}_{\mathcal{I},\mathsf{Sim}})) \in \mathcal{S}$ with $\mathsf{time}_{\mathsf{Sim}}(\mathcal{D}(\mathsf{Ideal}_{\mathcal{I},\mathsf{Sim}})) \in \mathcal{S}$, since $\mathcal{I}$ is unaffected, and $\mathcal{D}$ is w.l.o.g. a priori PPT.

Definition 5.5.2 can be extended to proof systems with unbounded provers, but technical artefacts can arise, see Remark C.5.21.

Remark 5.5.6 ("Environmental" distinguishing: Why $\mathcal{I}$ outputs *state*). In Definition 5.5.2, we allow $\mathcal{I}$ to output *state*, effectively making $(\mathcal{I}, \mathcal{D})$ into a stateful distinguishing "environment". Viewing Sim and P as oracles, this corresponds to oracle indistinguishability. Without this, the security does not obviously help when used as a subprotocol, since a protocol is effectively a (stateful) distinguishing environment. Definition 5.5.2 is discussed in-depth in Appendix C.5.4.1. Here, we only note that in the *non-uniform* classical PPT setting, it coincides with the standard definition.

Remark 5.5.7. We seldom mention non-uniform zero-knowledge formulations in the rest of this work. Our definitions, constructions and proofs make timed bb-rw use of the adversary, and therefore apply in the non-uniform setting without change.

5.5.2. Application to Graph 3-Colouring

To exemplify the setting, the technical challenges, and our techniques, we use the constant-round zero-knowledge proof of Goldreich and Kahan [GK96] as a worked example. We only prove zero-knowledge, as completeness and soundness are unconditional. Formal definitions of commitment schemes are in Appendix C.2. We assume *left-or-right (LR) oracles* in the hiding experiment for commitment schemes. Intuitively, we assume a built-in hybrid argument. (Security against CEPT adversaries follows from security against PPT adversaries by a simple truncation argument.)

5.5.2.1. The Protocol

We recall $\mathsf{G3C}_{\mathsf{GK}}$ from Section 5.1.2. It requires two different commitments schemes; $\mathsf{Com}^{(\mathrm{H})}$ is perfectly hiding, $\mathsf{Com}^{(\mathrm{B})}$ is perfectly binding. See [GK96] for the exact requirements. We assume non-interactive commitments for simplicity. The common input is $G = (V, E)$ and the prover's witness is a 3-colouring $\psi \colon V \to \{0, 1, 2\}$.

(P0) P sends $ck_{hide} \leftarrow \mathsf{Setup}^{(\mathrm{H})}(\lambda)$. ($ck_{bind} \leftarrow \mathsf{Setup}^{(\mathrm{B})}(\lambda)$ is deterministic.)

(V0) V randomly picks challenge edges $e_i \leftarrow E$ for $i = 1, \dots, N = \lambda \cdot \mathrm{card}(E)$, commits to them as $c_i^e = \mathsf{Com}^{(\mathrm{H})}(ck_{hide}, e_i)$, and sends all $\{c_i^e\}_{i=1,\dots,N}$.

(P1) P picks randomized colourings ψ_i for all $i = 1, \dots, N$ and commits to all node colours for all graphs in (sets of) commitments $\{\{c_{i,j}^{\psi}\}_{j \in V}\}_{i=1,\dots,N}$ using $\mathsf{Com}^{(\mathrm{B})}$. P sends all $c_{i,j}^{\psi}$ to V.

(V1) V opens the commitments c_i^e to e_i for all $i = 1, \dots, N$.

(P2) P aborts if any opening is invalid or $e_i \notin E$ for some i. Otherwise, for all iterations $i = 1, \dots, N$, P opens the commitments $c_{i,a}^{\psi}$, $c_{i,b}^{\psi}$ for the colours of the nodes of edge $e_i = (a, b)$ in repetition i.

(V2) V aborts iff any opening is invalid, any edge not correctly coloured, or if ck_{hide} is bad. Otherwise, V accepts.

In [GK96], delaying the check of ck_{hide} to the end of the protocol weakens the requirements on VfyCK, as the verifier may learn setup randomness of ck_{hide} at that point. But this is irrelevant for zero-knowledge.

5.5.2.2. Proof of Zero-Knowledge

Our goal is to show the following lemma.

Lemma 5.5.8. *Suppose* $\mathsf{Com}^{(\mathrm{H})}$ *and* $\mathsf{Com}^{(\mathrm{B})}$ *are a priori PPT algorithms. Then protocol* $\mathrm{G3C_{GK}}$ *in Section 5.5.2.1 is zero-knowledge against CEPT adversaries with a bb-rw CEPT simulator. Let* $(\mathcal{I}, \mathsf{V}^*)$ *be a CEPT adversary and suppose* $T := \mathsf{time}_{\mathsf{P}+\mathsf{V}^*}(\mathsf{Real}_{\mathcal{I},\mathsf{V}^*})$ *is* (t, ε)*-time. Then* Sim *handles* $(\mathcal{I}, \mathsf{V}^*)$ *in virtually expected time* $(t', 2\varepsilon + \varepsilon')$*. Here* ε' *stems from an advantage against the hiding property of* $\mathsf{Com}^{(\mathrm{B})}$*, hence* ε' *negligible. If the time to compute a commitment depends only on the message length, then* t' *is roughly* $2t$*.*

Our proof differs from that in [GK96] on two accounts: First, we do not use the runtime normalization procedure in [GK96]. This is because a negligible deviation from EPT is absorbed into the CEPT virtuality, namely ε'. Second, we handle designated CEPT adversaries. In particular, the runtime classes of simulator and adversary coincide. We first prove the result for *perfect* EPT adversaries.

Lemma 5.5.9. *The claims in Lemma 5.5.8 hold if* $T \in \mathcal{EPT}$*, i.e.* $\varepsilon = 0$*.*

Proof sketch. We proceed in game hops. The initial game being $\mathsf{Real}_{\mathcal{I},\mathsf{V}^*}$ and the final game being $\mathsf{Ideal}_{\mathcal{I},\mathsf{Sim}}$. We consider (timed) bb-rw simulation.

Game G_0 is the real protocol. The output is the verifier's output and *state* (from $\mathcal{I}$). From now on, we ignore the *state* output, since no game hop affects it.

Game G_1**:** If the verifier opens the commitments in 5.5.2.1 correctly, the game repeatedly rewinds it to 5.5.2.1 using fresh prover randomness, until it obtains a second run where V^* unveils the commitments correctly (in 5.5.2.1). The output is V^*'s output at the end of this second successful run. If the verifier failed in the first run, the protocol proceeds as usual. The outputs of G_1 and G_0 are identically distributed. It can be shown that this modification preserves (perfect) EPT of the overall game, i.e. G_1 is perfect EPT. More precisely, the (virtually) expected time is about $2t$ (plus emulation overhead). To see this, use that each iteration executes P's code with fresh randomness. For the analysis, condition to fix the randomness of everything but P; averaging over the randomness of $\mathcal{I}$, V^* (and $\mathcal{D}$), then extends the reasoning again. Since bb-rw-access fixes the randomness of V^* between rewinds, the probability that V^* opens the commitment in step 5.5.2.1 is p in each (independent) try. Hence, the number of rewinds is distributed geometrically, and $1 + p\sum_{i=1}^{\infty} i \cdot p(1-p)^{i-1} = 2$ is the expected number of overall iterations (including the first try). Consequently, the expected runtime doubles at most.[44]

Game G_2**:** Test if both (valid) openings of V^*'s commitments in 5.5.2.1 open to the same value. Else, G_2 outputs `ambig`, indicating equivocation of the commitment. This modification hardly affects the runtime, so it is still bounded roughly by $2t$. The probability for `ambig` is negligible, since one can (trivially) reduce to an adversary against the binding property of $\mathsf{Com}^{(\mathrm{B})}$. That is, there is an adversary $\mathcal{B}$ such that $|\Pr[\mathcal{D}(\mathrm{out}(\mathrm{G}_2) = 1)] - \Pr[\mathcal{D}(\mathrm{out}(\mathrm{G}_1) = 1)]| = \mathrm{Adv}^{\mathrm{bind}}_{\mathsf{Com}^{(\mathrm{B})}}(\mathcal{B})$.

[44] Formally arguing that the expected time is bounded by $2t$ is a bit more technical than for strict time bounds. But it follows easily from the independence of the iterations (due to fresh prover randomness), and the fact that t, in particular, upper bounds the expected time per iteration. In detail: Observe that each iteration has the same runtime distribution T'. Let T denote the random variable for a complete run (without rewinding) and let T'_i be the random variable denoting the i-th (hypothetical) iteration for $i \in \mathbb{N}$, where we also define iterations which never happen (due to an earlier success). Then clearly $\mathbb{E}[T'_i] = \mathbb{E}[T'] \le t$. Let I be random variable denoting the number of iterations. Then the total time is $\mathbb{E}[(T - T_1) + \sum_{i=1}^{I} T'_i] = \mathbb{E}[T] + \mathbb{E}[I-1]\mathbb{E}[T'_i] \le 2t$.

In **Game** G_3, the initial commitments (in 5.5.2.1) to 3-colourings are replaced with commitments to 0. These commitments are never opened. Thus, we can reduce distinguishing Games 2 and 3 to breaking the hiding property of $\mathsf{Com}^{(B)}$ modelled as left-or-right indistinguishability. More precisely, the reduction constructs real resp. all-zero colourings, and uses the *LR-challenge commitment oracle* $\mathcal{O}_b$ which receives two messages (m_0, m_1) and commits to m_b. Use m_0 to commit to the real colouring *(left)*, whereas m_1 is the all-zero colouring *(right)*. The modification of G_2 to "oracle committing" yields an EPT Game $G_{2'}$ (instantiated with $\mathcal{O}_0$). The modification of G_3 to $G_{3'}$ (with $\mathcal{O}_1$) is CEPT. This follows immediately from the standard reduction, because Games $G_{2'}$ and $G_{3'}$ differ only in their oracle, and the case of $\mathcal{O}_0$ is EPT. More precisely, the standard reduction applied to $\mathcal{O}_0$ and $\mathcal{O}_1$ yields an adversary $\mathcal{B}$ such that $|\Pr[\mathcal{D}(\mathsf{out}(G_2)) = 1] - \Pr[\mathcal{D}(\mathsf{out}(G_1)) = 1]| \geq \frac{1}{4}\,\mathrm{Adv}^{\mathrm{hide}}_{\mathsf{Com}^{(B)}}(\mathcal{B})$ infinitely often, assuming $\mathcal{B}$ has non-negligible advantage.

Consequently, Game $G_{3'}$ is efficient with (oracle) runtime $T_{3'} \overset{c}{\approx} T_{2'}$, and the output distributions of Games $G_{2'}$ and $G_{3'}$ are indistinguishable. Finally, note that Game G_3 and $G_{3'}$ only differ by (not) using oracle calls. Incorporating these oracles does not affect CEPT (as $\mathcal{O}_1$ is an *a priori* PPT oracle). Thus, G_3 is efficient (i.e. CEPT) as well. Assuming the time to compute a commitment depends only on the message length, a precise analysis shows, that the (virtually) expected time is affected negligibly (up to machine model artefacts).

In **Game** G_4, the commitments in the reiterations of 5.5.2.1 are replaced by commitments to pseudo-colourings for each e_i, that is, at the challenge edge e_i, two random different colours are picked, and all other colours are set to 0. If V^* equivocates, the game outputs `ambig`. The argument for efficiency and indistinguishability of outputs is analogous to the step from Game G_2 to Game G_3. It reduces all replacements to the hiding property in a single step. This is possible since our definition of hiding is left-or-right oracle indistinguishability with an arbitrary number of challenge commitments. As before, a precise analysis shows that the (virtually) expected time is affected negligibly.

All in all, if G_0 runs in (virtually) expected time t, then G_4 runs in expected time about $2t$, ignoring the overhead introduced by bb-rw emulation, etc. Moreover, the output is indistinguishable, i.e. $|\Pr[\mathcal{D}(\mathsf{out}(G_4)) = 1] - \Pr[\mathcal{D}(\mathsf{out}(G_0))] = 1| \leq \mathsf{negl}$.

The simulator is defined as in G_4: It makes a first test-run with an all-zeroes colouring. If the verifier does not open its challenge commitment in 5.5.2.1, Sim aborts (like the real prover in 5.5.2.1). Otherwise, it rewinds V^* (and uses pseudo-colourings) until V^* opens the challenge commitment again, and outputs the verifier's final output of this run (or `ambig`). (To prevent non-halting executions, we may abort after, e.g., $2^{2\lambda}$ steps. But this is not necessary for our results.) □

We point out some important parts of the proof: First, in Game G_1, rewinding and its preservation of EPT is unconditional. That is, rewinding is separated from the computational steps happening after it. Second, since the simulator's time per iteration is roughly that of prover and verifier, the total simulation time is CEPT (and roughly virtually expected $2t$). Third, with size-guarded security (Appendix C.5.4.3), we could have argued efficiency much simpler and coarser. It would suffice if the runtime per rewind is polynomial in the input size (not counting V^*).

There is only one obstacle to extend our result to CEPT adversaries. It is not obvious, whether the introduction of rewinding in G_1 preserves CEPT. Fortunately, this is quite simple to see: The probability that a certain commitment is sent in 5.5.2.1 increases, since the verifier is rewound and many commitments may be tried. However, the probability only increases by a factor of 2. Thus, "bad" queries are only twice as likely as before.

More concretely, using Lemma 5.3.12, we obtain a $\mathcal{I}'$ and $\mathcal{O}'$ which output `timeout` in case of "bad" queries. By the above claim, the probability for `timeout` at most doubles. Thus, the virtuality of G_1 is at most twice that of G_0, (and the virtually expected runtime is roughly doubled as well). Hence, G_1 is CEPT. We show this in more detail in the following proof.

Proof sketch of Lemma 5.5.8. G_0 **to** G_1**:** Fix the first message ck_{hide} of P to bbrw(V*) and the randomness of V* (which is fixed since we consider a bb-rw oracle). Let $p_b(c)$ be the probability, that in protocol step 5.5.2.1 G_b sends $c = \{\{c^{\psi}_{i,j}\}_{j\in V}\}_{i=1,\dots,N}$ to bbrw(V*) at least once. (For G_0, also at most once. But rewinding in G_1 increases the chances.) Let γ_i denote the i-th query sent in step 5.5.2.1 (or $\perp$ if none was sent), let the random variable I denote the total number of queries. Then

$$\begin{aligned} p_1(c) &= \Pr[\exists j \le i\colon \gamma_j = c \wedge I \le j] \le \sum_{i=1}^{\infty} \Pr[I \ge i \wedge \gamma_i = c] \\ &= \sum_{i=1}^{\infty} \Pr[I \ge i] \Pr[\gamma_i = c \mid I \ge i] \le \sum_{i=1}^{\infty} \Pr[I \ge i] \cdot p_0(c) = \mathbb{E}[I] \cdot p_0(c). \end{aligned}$$

In the penultimate equality, we use that, for any fixed i, γ_i is a fresh random commitment (or never sampled, if $I < i$). As argued before, $\mathbb{E}[I] = 2$, hence $p_1(c) \le 2p_0(c)$. Thus, the probability $p_1(c)$ for G_1 to issue query c is at most twice that of G_0. By averaging over first messages c (according to prover randomness), the derivation extends to our setting of interest, where c is chosen randomly by P. Next, we conclude from this, that the virtuality at most doubles.

By an application of Lemma 5.3.12, we can assume a perfect EPT V′ derived from V*, i.e. $\mathsf{time}_{\mathrm{V}'}(\mathrm{G}_0)$ is EPT. We can then use V′ the transition of from Game G_0 to G_1. Recall that V* and V′ are equal until `timeout` by construction.

Denote by G_0' the modification of G_0 which uses V′ instead of V*, and let G_0' immediately output `timeout` if V′ does. Then $\mathsf{time}_{\mathrm{V}'}(\mathrm{G}_0')$ is EPT by construction, and essentially equals the virtual expected time of $\mathsf{time}_{\mathrm{V}^*}(\mathrm{G}_0)$. The statistical distance $\Delta(\mathrm{G}_0, \mathrm{G}_0')$ is exactly the probability that V′ outputs `timeout`. Let G_1' be defined analogously to G_0'.

Let `timeout`(*query*) be 1 if *query* causes a `timeout` and 0 otherwise. Then

$$\begin{aligned} \Pr_{\mathrm{G}_1'}[\mathtt{timeout}] &= \sum_{query} \mathtt{timeout}(query) \cdot p_1(query) \\ &\le 2 \sum_{query} \mathtt{timeout}(query) \cdot p_0(query) = 2 \cdot \Pr_{\mathrm{G}_0'}[\mathtt{timeout}]. \end{aligned}$$

Since the probability for `timeout` bounds the virtuality if we use V* instead of V′, this shows that G_1 is CEPT, with virtuality 2ε. If G_0 always halts, the outputs of G_1 and G_0 are identically distributed. In general, the statistical distance is (at most) $2 \cdot \Pr[\mathrm{G}_0 = \mathtt{nohalt}]$; this follows as for virtuality, which must encompass the probability of non-halting executions. Conditioned on halting executions, the distributions G_0 and G_1 are identical. The transition to G_2 now relies on the standard reduction, all other steps of Lemma 5.5.9 apply literally. □

We abstract the above proof strategy in Section 5.6, to cover a large class of proof systems.

Remark 5.5.10. With an analogous proof, one finds that the simulator in [GK96] is also a CEPT simulator. Its advantage is, that it handles adversaries which are *a priori PPT*, as well as *EPT w.r.t. any reset attack* [Gol10], without introducing any "virtuality", i.e. the simulation is EPT. On the other hand, it increases virtuality of CEPT adversaries by a larger factor.

5.5.3. Sequential Composition of Zero-Knowledge

The formulation of sequential security is not merely sequential *repetition*, but considers adaptive choices of inputs. With this, our notion and proof is very close to modular sequential composition for SFE.

5.5.3.1. Security Definition

To model adaptive inputs, we replace the input generator $\mathcal{I}$ by an "environment" $\mathcal{E}$. This "environment" $\mathcal{E}$ provides all inputs for the protocol, but does not participate in the protocol execution itself, it only learns the participants final outputs. This definition of sequential composition allows adaptive sequential executions.

Informally, our definition of sequential zero-knowledge can be summarized as follows: Instead of indistinguishability of $\langle \mathsf{P}, \mathsf{V}^* \rangle$ and $\mathsf{Sim}(\mathsf{V}^*)$, we assume indistinguishability of $\mathsf{rep}(\langle \mathsf{P}, \mathsf{V}^* \rangle)$ and $\mathsf{rep}(\mathsf{Sim}(\mathsf{V}^*))$, i.e. indistinguishability under repeated trials (Definition 5.2.15).

Definition 5.5.11 (Sequential zero-knowledge). Let $\mathcal{T}, \mathcal{S} \in \{\mathcal{PPT}, \mathcal{CPPT}, \mathcal{EPT}, \mathcal{CEPT}\}$. Let (P, V) be an argument system. A **universal simulator** Sim takes as input $(\mathbb{x}, \mathrm{code}(\mathsf{V}^*), aux)$ and simulates V^*'s output. Let $(\mathcal{E}, \mathsf{V}^*, \mathcal{D})$ be an adversarial environment $\mathcal{E}$ and an adversarial verifier $\mathsf{V}^*$and a distinguisher $\mathcal{D}$. The environment is given access one of two oracles $\mathcal{O}_{\mathsf{P}}$, $\mathcal{O}_{\mathsf{Sim}}$, which take as input $(\mathbb{x}, \mathbb{w}, aux)$ and

- $\mathcal{O}_{\mathsf{P}}(\mathbb{x}, \mathbb{w}, aux)$ returns $\mathsf{out}_{\mathsf{V}^*}\langle \mathsf{P}(\mathbb{x}, \mathbb{w}), \mathsf{V}^*(aux)\rangle$. $\qquad (\mathcal{O}_{\mathsf{P}} \mathrel{\widehat{=}} \mathsf{rep}(\langle \mathsf{P}(\,\cdot\,), \,\cdot\, \rangle))$
- $\mathcal{O}_{\mathsf{Sim}}(\mathbb{x}, \mathbb{w}, aux)$ returns $\mathsf{Sim}(\mathbb{x}, \mathrm{code}(\mathsf{V}^*), aux)$. $\qquad (\mathcal{O}_{\mathsf{Sim}} \mathrel{\widehat{=}} \mathsf{rep}(\mathsf{Sim}(\,\cdot\,)))$

We assume that both oracles reject (say with $\bot$) if $(\mathbb{x}, \mathbb{w}) \notin \mathcal{R}$. We consider two executions, a real and an ideal one, defined by:

$$\begin{aligned} \mathsf{Real}_{\mathcal{E},\mathsf{V}^*}(\lambda) &:= \mathsf{out}_{\mathcal{E}}\langle \mathcal{E}, \mathsf{rep}(\mathcal{O}_{\mathsf{P}})\rangle \\ \text{and} \quad \mathsf{Ideal}_{\mathcal{I},\mathsf{Sim}}(\lambda) &:= \mathsf{out}_{\mathcal{E}}\langle \mathcal{E}, \mathsf{rep}(\mathcal{O}_{\mathsf{Sim}})\rangle \end{aligned}$$

We define $\mathsf{Real}_{\mathcal{E},\mathsf{V}^*}(\lambda)$ to be the execution of $(\mathcal{E}, \mathsf{V}^*)$ with $\mathcal{O}_{\mathsf{P}}$, and $\mathsf{Ideal}_{\mathcal{E},\mathsf{Sim}}(\lambda)$ the execution with $\mathcal{O}_{\mathsf{Sim}}$. The distinguishing advantage of $(\mathcal{E}, \mathsf{V}^*, \mathcal{D})$ is

$$\mathrm{Adv}^{\mathrm{zk}}_{\mathcal{E},\mathsf{V}^*,\mathcal{D}}(\lambda) := |\Pr[\mathcal{D}(\mathsf{Real}_{\mathcal{E},\mathsf{V}^*}(\lambda)) = 1] - \Pr[\mathcal{D}(\mathsf{Ideal}_{\mathcal{E},\mathsf{Sim}}(\lambda)) = 1]|.$$

A (designated) adversary $(\mathcal{E}, \mathsf{V}^*, \mathcal{D})$ is $\mathcal{T}$-time if $\mathsf{time}_{\mathcal{E}+\mathsf{P}+\mathsf{V}^*+\mathcal{D}}(\mathcal{D}(\mathsf{Real}_{\mathcal{E},\mathsf{V}^*})) \in \mathcal{T}$.

The argument system is **(uniform) sequential zero-knowledge** against $\mathcal{T}$-time adversaries w.r.t. $\mathcal{S}$-time Sim, if for any $\mathcal{T}$-time adversary $(\mathcal{E}, \mathsf{V}^*)$:

- $\mathsf{time}_{\mathcal{E}+\mathsf{Sim}+\mathcal{D}}(\mathcal{D}(\mathsf{Ideal}_{\mathcal{E},\mathsf{Sim}})) \in \mathcal{S}$, that is, $\mathsf{rep}(\mathcal{O}_{\mathsf{Sim}(\mathrm{code}(\mathcal{A}))})$ is weakly $(\mathcal{T}, \mathcal{S})$-efficient relative to $\mathsf{rep}(\mathcal{O}_{\langle \mathsf{P},\mathsf{V}^*\rangle})$.
- $\mathrm{Adv}^{\mathrm{zk}}_{\mathcal{E},\mathsf{V}^*,\mathcal{D}}(\lambda)$ is negligible

We also say that protocols with sequential zero-knowledge simulators *compose sequentially.* We dropped the input generator $\mathcal{I}$, since its complexity class is the same as that of the environment $\mathcal{E}$, For clarity, we did not "include" $\mathcal{D}$ in $\mathcal{E}$, although the resulting definition would be equivalent.

A few remarks are in order. One could fix the universal adversary $\mathcal{V}_{\mathrm{univ}}$ (which executes a given program) as V^* in Definition 5.5.11. For compatibility with Definition 5.5.2, we allow any V^*.

Remark 5.5.12. Sequential zero-knowledge where $\mathcal{E}$ is restricted to a single query is equivalent to auxiliary input zero-knowledge. This is easily seen since Definition 5.5.2 allows $\mathcal{I}$ to pass *state* to $\mathcal{D}$.

Caution 5.5.13. Taken literally, Definition 5.5.11 is unsuitable for expected time. Inefficiencies similar to the setting of bb-rw oracles arise. However, as with rewinding strategies, we use the usual convenient notation. We leave implicit, that an efficient implementation which is logically equivalent is easily derived.[45] In Definition 5.5.11, passing the state of V^* via *aux* runs into these problems. For simplicity, assume *aux* is shared memory between $\mathcal{E}$ and V^*.

5.5.3.2. Sequential Composition Lemma

We are now ready to state and prove the sequential composition lemma for zero-knowledge.

Lemma 5.5.14 (Sequential composition lemma)**.** *Let* (P, V) *be an argument system. Suppose* Sim *is a simulator for auxiliary input zero-knowledge (which handles CEPT adversaries in CEPT). Then* (P, V) *is sequential zero-knowledge (with the same simulator* Sim *which also handles CEPT adversaries against sequential zero-knowledge in CEPT).*

The proof is an almost trivial consequence of the hybrid lemma.

Proof. Let $(\mathcal{E}, \mathsf{V}^*, \mathcal{D})$ be a CEPT adversary. Let $\mathcal{O}_{\mathsf{P}}(\mathbb{x}, \mathbb{w}, aux)$ and $\mathcal{O}_{\mathsf{Sim}}(\mathbb{x}, \mathbb{w}, aux)$ be as in Definition 5.5.11. By definition,

$$\mathsf{Real}_{\mathcal{E},\mathsf{V}^*}(\lambda) = \mathsf{out}_{\mathcal{E}}\langle \mathcal{E}, \mathsf{rep}(\mathcal{O}_{\mathsf{P}})\rangle \qquad \text{and} \qquad \mathsf{Ideal}_{\mathcal{G},\mathsf{Sim}}(\lambda) = \mathsf{out}_{\mathcal{E}}\langle \mathcal{E}, \mathsf{rep}(\mathcal{O}_{\mathsf{Sim}})\rangle$$

Define a distinguisher $\mathcal{A}$ for $\mathsf{rep}(\mathcal{O}_{\mathsf{P}})$ and $\mathsf{rep}(\mathcal{O}_{\mathsf{Sim}})$ as $\mathcal{D}(\mathsf{out}_{\mathcal{E}}\langle \mathcal{E}, \mathsf{rep}(\mathcal{O})\rangle)$. Now, we are in the usual setting of oracle (in)distinguishability. Since Sim is an auxiliary input zero-knowledge simulator for (P, V), we have that $\mathcal{O}_{\mathsf{Sim}}$ is weakly efficient relative to $\mathcal{O}_{\mathsf{P}}$ and that $\mathcal{O}_{\mathsf{P}} \overset{c}{\approx} \mathcal{O}_{\mathsf{Sim}}$. Thus, the hybrid lemma for CEPT, Lemma 5.4.7, is applicable. Hence $\mathsf{rep}(\mathcal{O}_{\mathsf{P}})$ is weakly relative efficient to $\mathsf{rep}(\mathcal{O}_{\mathsf{Sim}})$ and $\mathsf{rep}(\mathcal{O}_{\mathsf{P}}) \overset{c}{\approx} \mathsf{rep}(\mathcal{O}_{\mathsf{Sim}})$. This concludes the proof. ☐

5.6. Benign Simulation

In this section, we define benign simulation. This abstracts the proof strategy for $\mathrm{G3C}_{\mathrm{GK}}$ in Section 5.5.2. Namely, we define *rewinding strategies* to abstract the rewinding step, and we define "*simple assumption*" to abstract the left-right hiding and binding property of the commitment. Put together, we define benign simulators as simulators which have a proof of security analogous to the one of $\mathrm{G3C}_{\mathrm{GK}}$.

[45] The problem is that passing around state is extremely wasteful, and involves copying the state to and from message interfaces. Generally speaking, almost anything, which does not go over a "real" network, should not be passed by copying. This can be solved in any number of ways. E.g. allow shared memory/tapes between machines, or introduce an additional interactive machine which represents that shared tape, and pass around (interface) access to memory/machine, and so on. It should be evident that it is easy but tedious to formalize this.

5.6.1. Rewinding Strategies

Rewinding strategies encapsulate the rewinding schedule of a simulator. Unlike simulators, their properties are unconditional.

Reminder (Black-box queries). By abuse of notation, we typically write $\mathsf{A}^{\mathbb{O}}$ instead of $\mathsf{A}^{\mathsf{bbrw}(\mathbb{O})}$, if it is understood that A has bb-rw access to $\mathbb{O}$. Our presentation treats $\mathsf{bbrw}(\mathbb{O})$ as a NextMsg oracle, but it is understood that a *logical* query $(m_1, \dots, m_\ell)$ is implemented efficiently by a *short* handle to the state of bb-rw$(\mathbb{O})$ after processing $(m_1, \dots, m_{\ell-1})$, and the message m_ℓ in that state.

5.6.1.1. Definitions

Our definition of rewinding strategies is specialized for zero-knowledge, but it generalizes to other settings easily.

Definition 5.6.1. A **rewinding strategy** RWS for a proof system (P, V) is an oracle algorithm with timed bb-rw access to the (malicious deterministic) verifier V^*. The output of RWS is a state of $\mathsf{bbrw}(\mathbb{O})$ (or an abort message), which we denote by the (logical) query leading to it.

A rewinding strategy RWS has **runtime tightness** poly, if the following holds: Let be $(\mathcal{I}, \mathsf{V}^*)$ any adversary (modelled as a timeful oracle). Let

$$T := \mathsf{time}_{\mathsf{P}+\mathsf{V}^*}(\langle \mathsf{P}, \mathsf{V}^* \rangle_{\mathcal{I}}) \quad \text{and} \quad S := \mathsf{time}_{\mathsf{RWS}+\mathsf{V}^*}(\mathsf{RWS}^{\mathsf{V}^*(\mathbb{x}, aux)}(\mathbb{x}, \mathbb{w}))$$

with input distribution $\mathcal{I}$. Then $\mathbb{E}[S] \leq \mathsf{poly} \cdot \mathbb{E}[T]$ for all $(\mathcal{I}, \mathsf{V}^*)$.[46]

Equivalently, for *deterministic* timeful $\mathcal{I}$, i.e. any sequence $(\mathbb{x}_\lambda, \mathbb{w}_\lambda, aux_\lambda) \in \mathcal{R}$ and any *deterministic* timeful V^*, the analogous claim holds.

The notion of *runtime tightness* of RWS is strong and unconditional. Up to minor technical details, it is equivalent to the notion of "normal machine" implicit in [Gol10, Definition 6]. The equivalence of using probabilistic and deterministic adversaries follows easily: Certainly, probabilistic covers deterministic. For the converse, one uses the tightness bound poly and linearity of expectation.

Remark 5.6.2 (Preservation of EPT). It is clear that a rewinding strategy RWS with *polynomial* runtime tightness **preserves EPT**, i.e. in the setting of Definition 5.6.1, if $\mathsf{time}_{\mathsf{P}+\mathsf{V}^*}(\langle \mathsf{P}, \mathsf{V}^* \rangle_{\mathcal{I}}) \in \mathcal{EPT}$, then $\mathsf{time}_{\mathsf{RWS}+\mathsf{V}^*}(\mathsf{RWS}^{\mathsf{V}^*(\mathbb{x}, aux)}) \in \mathcal{EPT}$.

Before we tackle preservation of CEPT, we introduce more parameters of rewinding strategies.

Definition 5.6.3 (Properties of rewinding strategies.). Let (P, V) be a proof system and RWS a rewinding strategy. Let $\mathcal{LQ}$ be the set of all possible (logical) queries. Suppose V^* is some (malicious) deterministic verifier (as a timeful oracle). Let λ, $(\mathbb{x}, \mathbb{w})$, aux be inputs. Let $query \in \mathcal{LQ}$ be a (logical) query to $\mathsf{bbrw}(\mathsf{V}^*)$. Let $\mathsf{pr}_{\mathrm{real}}(query)$ be the probability that, in a real interaction $\langle \mathsf{P}(\mathbb{x}, \mathbb{w}), \mathsf{V}^*(\mathbb{x}, aux) \rangle$, the prover queries *query*, that is[47]

$$\mathsf{pr}_{\mathrm{real}}(query) = \Pr[query \in \mathsf{qseq}_{\mathsf{P}}(\langle \mathsf{P}(\mathbb{x}, \mathbb{w}), \mathsf{V}^*(\mathbb{x}, aux) \rangle)].$$

[46] We define that $\infty \leq \infty$.

[47] By abuse of notation, we write $\mathsf{qseq}_{\mathsf{P}}(\langle \mathsf{P}(\mathbb{x}, \mathbb{w}), \mathsf{V}^*(\mathbb{x}, aux) \rangle)$ for the sequence of *logical* queries to $\mathsf{NextMsg}_{\mathsf{V}^*}$, i.e. $\mathsf{bbrw}(\mathsf{V}^*)$. Formally, $\mathsf{qseq}_{\mathsf{P}}(\dots)$ is the sequence of message sent by P, (and P does not treat V^* as $\mathsf{bbrw}(\mathsf{V}^*)$). So actually, we consider the sequence of prefixes of $\mathsf{qseq}_{\mathsf{P}}(\dots)$, which correspond to the logical queries to $\mathsf{bbrw}(\mathsf{V}^*)$ which result in the same execution as $\langle \mathsf{P}, \mathsf{V} \rangle$.

Let $\mathsf{pr}_{\mathsf{rws}}(query)$ be the probability, that $\mathsf{RWS}^{\mathsf{V}^*}(\mathbb{x}, \mathbb{w})$ queries *query*, that is

$$\mathsf{pr}_{\mathsf{rws}}(query) = \Pr[query \in \mathsf{qseq}_{\mathsf{RWS}}(\mathsf{RWS}^{\mathsf{V}^*(\mathbb{x}, aux)}(\mathbb{x}, \mathbb{w}))].$$

We say a rewinding strategy RWS has **probability tightness** $\mathsf{poly}_{\mathsf{pr}}(\lambda)$ if

$$\mathsf{pr}_{\mathsf{rws}}(query) \leq \mathsf{poly}_{\mathsf{pr}}(\lambda) \cdot \mathsf{pr}_{\mathsf{real}}(query)$$

for all queries $query \in \mathcal{LQ}$. (In other words: $\rho_{\mathsf{sup}}(\mathsf{pr}_{\mathsf{rws}}/\mathsf{pr}_{\mathsf{real}}) \leq \mathsf{poly}_{\mathsf{pr}}$.)

The output skew of RWS for an execution with $(\mathcal{I}, \mathsf{V}^*)$ (where $(\mathbb{x}, \mathbb{w}, aux) \leftarrow \mathcal{I}(\lambda)$) is similarly defined by the ratio $\rho_{\mathsf{sup}}(Y/X)$ of the output distributions, Y of RWS and X of P running with V^* on input sampled by $\mathcal{I}$.[48] We say RWS has **output skew** $\delta = \delta(\lambda)$, if for every (deterministic) $(\mathcal{I}, \mathsf{V}^*)$ which halts (with probability 1), the output skew for $(\mathcal{I}, \mathsf{V}^*)$ is at most $1 + \delta(\lambda)$. We say RWS has **perfect output (distribution)** if for all $(\mathcal{I}, \mathsf{V}^*)$ which always halt with probability 1 the output of RWS is distributed identically (that is, $\delta = 0$) to the real execution.

We note that the properties in Definition 5.6.3 are unconditional. Also, non-halting executions can affect the output distribution, as they will be encountered by RWS with higher probability than in the real execution, increasing the probability that RWS "outputs" `nohalt`. In any situation where *statistical* properties are good enough, one can assume that all algorithms halt (e.g. by truncation or modifying the machine model).

Finally, we define our notion of normality. The definition is similar to Goldreich's definition of normality in [Gol10].[49]

Definition 5.6.4 (Normal RWS). A rewinding strategy RWS is **normal** if it has polynomial runtime tightness, polynomial probability tightness, and perfect output distribution.

Perfect output distribution is vital for later use of RWS (as a part of security proofs). Negligible output skew would suffice, but natural rewinding strategies seem to satisfy perfect output skew, so we require that for simplicity.

5.6.1.2. Basic Results

Now, we state our main result for normal rewinding strategies.

Lemma 5.6.5 (Normal rewinding strategies preserve CEPT)**.** *Let* RWS *be a normal rewinding strategy for* (P, V). *Let* $(\mathcal{I}, \mathsf{V}^*)$ *be a CEPT adversary for zero-knowledge, that is* $\mathsf{time}_{\mathcal{I}+\mathsf{P}+\mathsf{V}^*}(\langle \mathsf{P}, \mathsf{V}^* \rangle_{\mathcal{I}}) \in \mathcal{CEPT}$. *Then* $\mathsf{time}_{\mathcal{I}+\mathsf{RWS}+\mathsf{V}^*}(\mathsf{RWS}^{\mathsf{V}^*(\mathbb{x}, aux)}(\mathbb{x}, \mathbb{w})) \in \mathcal{CEPT}$, *where* $(\mathbb{x}, \mathbb{w}, aux, state) \xleftarrow{\$} \mathcal{I}(\lambda)$.

More precisely, suppose $\mathsf{poly}_{\mathsf{time}}$ *is a runtime tightness and* $\mathsf{poly}_{\mathsf{virt}}$ *a probability tightness of* RWS *(against EPT adversaries). If* $\mathsf{time}_{\mathsf{P}+\mathsf{V}^*}(\langle \mathsf{P}, \mathsf{V}^* \rangle_{\mathcal{I}})$ *is virtually* (t, ε)*-time, then* $\mathsf{time}_{\mathsf{RWS}+\mathsf{V}^*}(\mathsf{RWS}^{\mathsf{V}^*})$ *is virtually* $(\mathsf{poly}_{\mathsf{time}} \cdot t, \mathsf{poly}_{\mathsf{virt}} \cdot \varepsilon)$*-time. In other words,* RWS *is efficient relative to* $\langle \mathsf{P}, \cdot \rangle$ *with runtime tightness* $(\mathsf{poly}_{\mathsf{time}}, \mathsf{poly}_{\mathsf{virt}})$.

[48] More correctly, X and Y are the distributions of the state of the timed bb-rw V^*.

[49] Goldreich remarks [Gol10, Footnote 24] that his notion of normality *of a simulator* is probably satisfied if the runtime analysis is *unconditional.* We separate the analysis into rewinding strategies and indistinguishability transitions, since our notion of runtime and efficiency of simulators is *not unconditional.* Disregarding this, the notions essentially coincide.

The proof exploits that "bad queries", which result in overly long runs of $\langle \mathsf{P}, \mathsf{V}^* \rangle_{\mathcal{G}}$ happen at most polynomially more often with RWS, due to normality. Since bad queries happen with probability ε, the claim follows. A detailed proof follows.

Proof. By Lemma 5.3.12, we know that there is a modification V' of V^* such that $\mathsf{time}_{\mathsf{V}'}(\langle \mathsf{P}, \mathsf{V}' \rangle_{\mathcal{G}})$ is EPT, where V' is a timeful oracle which aborts bad executions with `timeout`. By normality, also $\mathsf{time}_{\mathsf{V}'}(\mathsf{RWS}^{\mathsf{V}'})$ is EPT. We call a (logical) query $query = (m_1, \dots, m_n)$ to V' which returns `timeout` a *timeout query*. The probability that such a timeout query happens in a real execution with P is at most ε (by construction).

The only case where RWS encounters a difference between $(\mathcal{G}, \mathsf{V}^*)$ and $(\mathcal{G}, \mathsf{V}')$ is if RWS asks a timeout query, i.e. if V' returns `timeout`. By normality of RWS, the probability of asking a timeout query is only polynomially higher than the probability that P asks a timeout query. The latter is at most ε, hence the former is bounded by $\mathsf{poly}_{\mathsf{virt}} \cdot \varepsilon$. Thus, the runtime $\mathsf{time}_{\mathsf{RWS+V}^*}(\mathsf{RWS}^{\mathsf{V}^*})$ is CEPT with virtually expected time $(\mathsf{poly}_{\mathsf{time}} t, \mathsf{poly}_{\mathsf{virt}} \varepsilon)$. The claim for the total runtime follows analogously. □

We note that runtime tightness already implies probability tightness (see Remark C.5.30). However, the implied bounds are far from optimal. Following lemma is a simple way to get a tight(er) bound on probability tightness.

Lemma 5.6.6. *Let* RWS *be a rewinding strategy for* (P, V)*, and let* $(\mathcal{G}, \mathsf{V}^*)$ *be a timeful adversary. Let* $\mathcal{Q}_i \subseteq \mathsf{qseq}_{\mathsf{RWS}}(\mathsf{RWS}^{\mathsf{V}^*})$ *be the list of queries of length i from* RWS *to* $\mathsf{bbrw}(\mathsf{V}^*)$*; that is* $\mathcal{Q}_i$ *consists of queries* $(m_1, \dots, m_i)$*. Let* $Q_i = \mathsf{card}(\mathcal{Q}_i)$*. Note that* $\mathcal{Q}_i$ *and* Q_i *are random variables. Let* $Q = \sum_{i=0}^{\infty} Q_i$ *be the total number of queries. Suppose that for all adversaries,* $\mathbb{E}[Q_i] \leq M_i$ *for some* M_i*.*

Let $\mathsf{pr}_{\mathsf{rws}}(query)$ *resp.* $\mathsf{pr}_{\mathsf{real}}(query)$ *be the probability that* RWS *resp.* P *queries query, as in Definition 5.6.3. Write* $\mathcal{Q}_i[j]$ *for the j-th query in* $\mathcal{Q}_i$*. Suppose that for all i and all logical queries query of length i*

$$\forall j \in \mathbb{N}_0\colon \quad \Pr[query = \mathcal{Q}_i[j] \mid Q_i \geq j] \leq \mathsf{pr}_{\mathsf{real}}(query),$$

where the probability is over the randomness of RWS *and* P*. Then*

$$\mathsf{pr}_{\mathsf{rws}}(query) \leq M_i \cdot \mathsf{pr}_{\mathsf{real}}(query).$$

In particular, the probability tightness of RWS *is bounded by* $M = \max_i M_i$*.*

The basic idea behind Lemma 5.6.6 is that for any $(i-1)$-length history $m' = (m_1, \dots, m_{i-1})$, the probability that the prover queries m_i (conditioned on m') is identical to the probability that RWS queries m_i "conditioned on m'". The "conditioning RWS on m'" part needs a suitable definition. In special cases, e.g. "tree-based" rewinding strategies, this can be done hands on. Lemma 5.6.6 gives a general formalization of this idea (without needing to condition on some m').

It is often (almost) trivial to verify the conditions of Lemma 5.6.6. Moreover, we are not aware of (natural) rewinding strategies which do not satisfy normality, even outside the context of zero-knowledge.

Proof of Lemma 5.6.6. The proof is almost trivial. Consider the setting and notation of Lemma 5.6.6. Let *query* be a logical query of length i. We have

$$\Pr[\exists j\colon query = \mathcal{Q}_i[j]] \leq \sum_{j=0}^{\infty} \Pr[query = \mathcal{Q}_i[j]] = \sum_{j=0}^{\infty} \Pr[query = \mathcal{Q}_i[j] \mid Q_i \geq j] \Pr[Q_i \geq j],$$

by a union bound, and we have

$$\sum_{j=0}^{\infty} \Pr[query = \mathcal{Q}_i[j] \mid Q_i \geq j] \Pr[Q_i \geq j] \;=\; \sum_{j=0}^{\infty} \mathrm{pr}_{\mathrm{real}}(query) \Pr[Q_i \geq j] \;\leq\; \mathrm{pr}_{\mathrm{real}}(query) \cdot \mathbb{E}[Q_i].$$

by assumption (and by $\mathbb{E}[Q_i] = \sum_{j=0}^{\infty} \Pr[Q_i \geq j]$). □

The criterion in Lemma 5.6.6 is "global" and not "local", making it somewhat inconvenient. Instead of applying Lemma 5.6.6, it is often simple(r) to derive more precise bounds and directly prove normality.

Remark 5.6.7 (Partial RWS). A typical proof strategy for normality is to view RWS as a composition of (partial) strategies. For example, many rewinding strategies are "tree-based" and each layer corresponds to a (partial) rewinding strategy, which calls lower layers as substrategies. This approach lends itself to a simple and precise analysis of runtime tightness, probability tightness and "query tightness". For example, if calls to substrategies not skewed, probability tightness behaves multiplicatively. Checking normality like this relies on "local" properties, which by composition yield the "global" properties.

Remark 5.6.8. Halevi and Micali [HM98] define "valid distributions" [of transcripts] for extraction in the context of proofs of knowledge. Their definition requires that a polynomial number of total executions are made (with the extractor in the role of the verifier), and each execution has a transcript (i.e. queries) which is distributed like for an honest verifier. Separate runs may be stochastically dependent. Lemma 5.6.6 deals with partial transcripts, expected polynomially many executions, and probability tightness (not runtime), but is otherwise similar to [HM98].

5.6.1.3. Examples of Normal Rewinding Strategies

We give some examples for rewinding strategies which are normal. Most claims follows easily from their original efficiency analysis.

Example 5.6.9 (The classic cut-and-choose protocols). The classic protocols for graph 3-colouring, graph hamiltonicity, as well as graph-(non)-isomorphpism [GMW86; Blu86] use normal rewinding strategies.

Example 5.6.10 (Constant round zero-knowledge). Our motivating example [GK96], the simplification of Rosen [Ros04], and the proof of knowledge of Lindell [Lin13] have normal rewinding strategies.

Example 5.6.11 (Concurrent zero-knowledge). The concurrent zero-knowledge proof systems of Kilian and Petrank [KP01] and its variation [PTV14] also rely on normal rewinding strategies. Indeed, their strategy is strictly PPT (in oracle-excluded time).

Example 5.6.12 (Blum coin-toss). The simulator for the coin-toss protocol [Blu81; Lin17] also gives rise to normal rewinding strategies. It is strictly PPT (in oracle-excluded time).

5.6.2. Simple Assumptions and Repeated Trials

To obtain nice results, we want nice "base assumptions" to reduce security to. We call these "*simple assumptions*". For simplicity, we do not allow (shared) setups, such as a common random string, and are very restrictive w.r.t. the runtime of such oracles.

Definition 5.6.13 (PPTpa). A timeful oracle $\mathcal{O}$ is **a priori PPT per activation (PPTpa)**, if there is a polynomial poly such that every invocation of $\mathcal{O}$ has runtime bounded by $\mathsf{poly}(\lambda)$.

The property we need from a priori PPTpa is that, if a distinguisher yields an inefficient system, then the oracle is never to blame, i.e. even excluding its runtime, the system is inefficient. There are less strict efficiency notions which satisfy this as well, but PPTpa is sufficient for our purposes.

Definition 5.6.14 (Simple assumption). Let $\mathcal{C}_0$ and $\mathcal{C}_1$ be two oracles, induced by algorithms which are *a priori PPT per activation.* The assumption that $\mathcal{C}_0$ and $\mathcal{C}_1$ are indistinguishable (w.r.t. PPT adversaries) is called a **simple assumption**. We also say $\mathcal{C}_0$ and $\mathcal{C}_1$ form a simple assumption.

Example 5.6.15. Many assumptions are simple, for example one-way functions, trap-door one-way permutations, pseudorandom functions, hiding and binding properties of commitments, IND-CPA and IND-CCA security of public key encryption, and so on. Counterexamples are 1-more assumptions, e.g. the one-more RSA assumption. Knowledge assumptions are also not simple. Note that assumptions which can be reduced to simple assumptions need not be simple, e.g. soundness of (non-extractable) proof systems.

By definition, simple assumptions are essentially falsifiable assumptions [Nao03] as defined by Gentry and Wichs [GW11]. However, the (invisible) intent of simple assumptions is that they have a simple notion of repeated trials, and behave well in this setting. Since our primary setting is the plain model, simple assumptions are natural, but we stress that our techniques work for a much broader class of game-based assumptions, including non-falsifiable assumptions.[50]

Simple assumptions are secure under repeated trials against PPT (or CEPT) adversaries.

Lemma 5.6.16 (Hybrid lemma for simple assumptions)**.** *Let $\mathcal{C}_0$ and $\mathcal{C}_1$ be two oracles forming a simple assumption, and let $q = q(\lambda)$ where $q = \infty$ is allowed. Suppose $\mathcal{D}$ is a CEPT distinguisher for q-repeated trials, with $|\mathsf{Adv}^{dist}_{\mathcal{D},\mathsf{rep}_q(\mathcal{C}_0),\mathsf{rep}_q(\mathcal{C}_1)}| \geq \varepsilon = 1/\mathsf{poly}$ infinitely often. Suppose $\mathsf{time}_{\mathcal{D}}(\mathcal{D}^{\mathsf{rep}_q(\mathcal{C}_0)})$ is bounded by (t_0, v_0). Let $M(\lambda) \geq \min(q(\lambda), 4\varepsilon^{-1}t_0)$ be an (efficiently computable) polynomial upper bound. Then there is an a priori PPT distinguisher $\mathcal{A}$ with advantage at least $\frac{1}{M}(\frac{\varepsilon}{4} - v_0)$ infinitely often.*

This immediately yields:

Corollary 5.6.17. *Let $\mathcal{C}_0$ and $\mathcal{C}_1$ form a simple assumption, in particular, $\mathcal{C}_0 \overset{c}{\approx} \mathcal{C}_1$. Then $\mathsf{rep}(\mathcal{C}_0)$ and $\mathsf{rep}(\mathcal{C}_1)$ form a simple assumption, in particular, $\mathsf{rep}(\mathcal{C}_0) \overset{c}{\approx} \mathsf{rep}(\mathcal{C}_1)$.*

[50] Typical 1-more assumptions have a meaningful notion of security under repeated trials as well, but Definition 5.2.14 is too coarse to capture this, as it postulates independent instances. For example, given two 1-more-dlog oracles for a deterministic group generator, it is easy to win in one of the 1-more dlog instances; but by correlating the repeated oracles, one can also embed a 1-more-dlog challenge.

Proof of Lemma 5.6.16. First apply Corollary 5.4.2 to get an a priori PPT distinguisher $\mathcal{A}'$. Note that we treat distinguishing under repeated trials as distinguishing $\mathcal{O}_0^* = \mathsf{rep}(\mathcal{C}_0)$ and $\mathcal{O}_1^* = \mathsf{rep}(\mathcal{C}_1)$. Thus, we end up with advantage $\frac{\varepsilon}{4} - \nu_0$ and runtime bound roughly $4\varepsilon^{-1}t_0$, where (t_0, ν_0) is virtually expected time of $\mathcal{D}$ as in Lemma 5.4.1. In particular, $\mathcal{A}'$ can make at most $M(\lambda)$ queries.

Now, we rely on the efficient implementation of $\mathcal{C}_0$, $\mathcal{C}_1$ to implement the hybrid distinguisher. The claim follows from the (standard a priori) PPT) hybrid lemma. □

5.6.3. Benign Simulators

Our definition of a benign simulator abstracts the proof strategy for $\mathsf{G3C_{GK}}$. Before we give the definition, we demonstrate the idea.

Example 5.6.18 (Structure of the security reduction for $\mathsf{G3C_{GK}}$). Consider the protocol $\mathsf{G3C_{GK}}$ in Section 5.5.2.1 and the security proof in Section 5.5.2.2. Let $(\mathcal{I}, \mathsf{V}^*, \mathcal{D})$ be an adversary. Since the simulator cannot depend on $\mathcal{I}$ and $\mathcal{D}$, they are of no importance in the following. Indeed, they should be viewed as one entity, the "distinguisher", whereas V^* is the actual "attacker". Below, we omit the inputs $\mathbb{x}$, $\mathbb{w}$, *aux*.

Let $\mathsf{A}_0 = \mathsf{A}_0(\mathsf{V}^*)$ denote the algorithm $\mathsf{out}_{\mathsf{V}^*}\langle \mathsf{P}, \mathsf{V}^*\rangle$. Let $\widetilde{\mathsf{A}}_0 = \widetilde{\mathsf{A}}_0(\mathsf{V}^*)$ denote the algorithm which introduces all rewindings, as in Section 5.5.2.2, G_1. Moreover, $\widetilde{\mathsf{A}}_0$ makes any commitment computations into explicit calls to subroutines. (Let us call this *boxing*, and the act of "forgetting" subroutine calls *unboxing*.)

We note the following: For any V^*, $\mathsf{A}_0 \equiv \widetilde{\mathsf{A}}_1$ (i.e. they are perfectly indistinguishable), and if A_0 is efficient, then so is $\widetilde{\mathsf{A}}_0$. More concretely: For every $\mathcal{I}, \mathcal{D}$, if the completed system for A_0 is CEPT (i.e. with inputs sampled by $\mathcal{I}$ and with $\mathcal{D}$ applied to the output of A_0), so is the completed system for $\widetilde{\mathsf{A}}_0$.

Similarly, let $\mathsf{A}_1 \coloneqq \mathsf{Sim}(\mathsf{V}^*)$ and let $\widetilde{\mathsf{A}}_1 = \widetilde{\mathsf{A}}_1(\mathsf{V}^*)$ be the simulator with boxed calls to Com. Clearly, for any V^*, $\widetilde{\mathsf{A}}_1 \equiv \mathsf{A}_1$, and if $\widetilde{\mathsf{A}}_1$ is efficient (i.e. CEPT), so is A_1.

Consider the two indistinguishable oracles $\mathcal{O}_0$, $\mathcal{O}_1$, which represent the (repeated) binding and hiding experiments in the security proof, squeezed into one oracle. It is straightforward to define an (oracle) algorithm $\mathsf{R} = \mathsf{R}(\mathsf{V}^*)$, which encapsulates the reduction given in the games following G_1 in Section 5.5.2.2, such that for R, it holds that $\widetilde{\mathsf{A}}_0 \equiv \mathsf{R}^{\mathcal{O}_0}$ and $\mathsf{R}^{\mathcal{O}_1} \equiv \widetilde{\mathsf{A}}_1$. Moreover, $\mathsf{R}^{\mathcal{O}_0}$ is efficient if $\widetilde{\mathsf{A}}_0$ is. Furthermore, since $\mathcal{O}_0$ and $\mathcal{O}_1$ are indistinguishable, if $\mathsf{R}^{\mathcal{O}_0}$ is CEPT, so is $\mathsf{R}^{\mathcal{O}_1}$. (This step relies on CEPT and fails for EPT.)

Consequently, $\mathsf{Sim}(\mathsf{V}^*)$ is CEPT whenever $\langle \mathsf{P}, \mathsf{V}^*\rangle$ is CEPT, and $\mathsf{Sim}(\mathsf{V}^*)$ and $\langle \mathsf{P}, \mathsf{V}^*\rangle$ are computationally indistinguishable. Pictorially, the security proof worked as follows:

$$\mathsf{A}_0 \xrightarrow[e]{\equiv} \widetilde{\mathsf{A}}_1 \xrightarrow[e]{\equiv} \mathsf{R}^{\mathcal{O}_0} \overset{c}{\approx} \mathsf{R}^{\mathcal{O}_1} \xrightarrow[e]{\equiv} \widetilde{\mathsf{A}}_1 \xrightarrow[e]{\equiv} \mathsf{A}_1,$$

where $\mathsf{A} \xrightarrow[e]{\equiv} \mathsf{B}$ denotes that A and B are perfectly indistinguishable and that if B is efficient (given V^*), so is A. More precisely, we have

$$\langle \mathsf{P}, \cdot\,\rangle \xrightarrow[e]{\equiv} \mathsf{RWS}(\,\cdot\,) \xrightarrow[e]{\equiv} \mathsf{R}^{\mathcal{O}_0}(\,\cdot\,) \overset{c}{\approx} \mathsf{R}^{\mathcal{O}_1}(\,\cdot\,) \xrightarrow[e]{\equiv} \widetilde{\mathsf{Sim}}(\,\cdot\,) \xrightarrow[e]{\equiv} \mathsf{Sim}(\,\cdot\,),$$

where we made explicit, that this construction is functional in the adversary (the missing argument denoted " · "). We also note that the intermediate steps ($\widetilde{\mathsf{A}}_0$, $\widetilde{\mathsf{A}}_1$, resp. RWS, $\widetilde{\mathsf{Sim}}$) can be omitted.

As a first step, we have define what a "reduction" is. Simple "reductions" are just connections of two interactive algorithms (which depend on $\mathscr{A}$) by an indistinguishability assumption. The name "reduction" is debatable, and the definition very restrictive, but sufficient for our purposes.

Definition 5.6.19 (Simple reductions). A **simple reduction** under an (implicit) simple assumption $(\mathcal{O}_0, \mathcal{O}_1)$ is an oracle algorithm R which expects expects access to an oracle $\mathcal{O}_b$ and $\mathrm{code}(\mathscr{A})$ as input. Given $\mathcal{O}$ and $\mathrm{code}(\mathscr{A})$, $\mathrm{R}^{\mathcal{O}_b}(\mathscr{A})$ implements (an interactive) algorithm.

Our definition of benign simulation requires a security proof as sketched in Example 5.6.18, and is basically an abstract formalization of that proof strategy. For completeness, we give a more traditional approach in Appendix C.6, which relies on indistinguishability of queries similar to [KL08]. We view both approaches as complementary: Our definition of benign simulation is *easily applicable* to typical protocols (and all of our examples), whereas the query-indistinguishability condition is something one can arguably expect from almost any simulator, which *broadens* the class of simulators which handle CEPT adversaries in CEPT. In any case, a bb-rw simulation with a normal rewinding strategy is assumed.

Definition 5.6.20 (Benign simulation). Let (P, V) be an argument system. Let Sim be a *(timed) bb-rw* simulator with **associated rewinding strategy** RWS and **associated simple reduction** R under simple assumption $(\mathcal{O}_0, \mathcal{O}_1)$. Moreover, the reduction $\mathrm{R}^{\mathcal{O}_b}(\mathsf{V}^*)$ has the interface of RWS, i.e. it expects $(\mathrm{code}(\mathsf{V}^*), \mathbb{x}, \mathbb{w}, aux)$. Suppose that, for any adversary V^*:

1. RWS is a *normal* rewinding strategy.
2. $\mathsf{RWS}^{\mathsf{V}^*} \equiv \mathrm{R}^{\mathcal{O}_0}(\mathsf{V}^*)$ and $\mathrm{R}^{\mathcal{O}_0}(\mathsf{V}^*)$ is efficient relative to $\mathsf{RWS}^{\mathsf{V}^*}$ with polynomial runtime tightness.
3. $\mathrm{R}^{\mathcal{O}_1}(\mathsf{V}^*) \equiv \mathsf{Sim}(\mathsf{V}^*)$ and $\mathsf{Sim}(\mathsf{V}^*)$ is efficient relative to $\mathrm{R}^{\mathcal{O}_1}(\mathsf{V}^*)$ with polynomial runtime tightness.
4. $\mathcal{O}_0$ and $\mathcal{O}_1$ form a simple assumption, and are indistinguishable, i.e. $\mathcal{O}_0 \overset{c}{\approx} \mathcal{O}_1$.

Then Sim is **benign** (under the assumption $\mathcal{O}_0 \overset{c}{\approx} \mathcal{O}_1$).

5.6.3.1. Iterated Benign Reductions

Our definition of benign allows only *one* "reduction step" using $\mathcal{O}_0 \overset{c}{\approx} \mathcal{O}_1$. Many security proofs can be squeezed into this setting. However, a simple relaxation is useful.

Definition 5.6.21 (Iterated benign). In the setting of Definition 5.6.20, we call Sim **iterated benign**, if there is a *constant* k and a sequence of "intermediate simulators" $\mathsf{Sim}_0, \dots, \mathsf{Sim}_k$, which expect as input $(\mathrm{code}(\mathscr{A}), \mathbb{x}, \mathbb{w}, aux)$ so that

1. $\mathsf{Sim}_0 \equiv \langle \mathsf{P}, \cdot \rangle$ and $\mathsf{Sim}_k \equiv \mathsf{Sim}$ (where Sim_k ignores $\mathbb{w}$).
2. Sim_i and Sim_{i+1} are related by a benign reduction (as in Definition 5.6.20, with oracles $\mathcal{O}_{i,b}$, $i = 1, \dots, k$, $b = 0, 1$.).

We stress that iterated benign only allows a *constant* number of "hops". The reason is that runtime may double for each hop, so superconstantly many "hops" *could* make the runtime explode. Thus, hybrid arguments must be put into the (simple) assumptions. Also note that RWS can be absorbed into R, and the relative efficiency requirement. While we the (possible) use of RWS explicit in Definition 5.6.20, we left it implicit in Definition 5.6.21.

5.6.3.2. Examples of (Iterated) Benign Simulators

All of our examples can be easily expressed via (iterated) benign simulators. We stress that hybrid arguments must be incorporated into the (simple) assumptions $\mathcal{C}_{i,0} \overset{c}{\approx} \mathcal{C}_{i,1}$.

Example 5.6.22. The classic, the constant round, and the concurrent zero-knowledge protocol examples [GMW86; Blu86; GK96; Ros04; Lin13; KP01; PTV14] from Section 5.6.1.3 have benign simulation.

5.6.3.3. Zero-Knowledge and Benign Simulation

We only give results for benign simulation. Extending these to iterated benign is straightforward and left to the reader.

Lemma 5.6.23. *Suppose* (P, V) *is an argument system. Let* Sim *be a benign simulator. Then* Sim *is a zero-knowledge simulator which handles CEPT adversaries (in CEPT).*

Proof. Suppose $(\mathcal{I}, \mathsf{V}^*, \mathcal{D})$ is an adversary which is CEPT in the real protocol. For brevity, whenever we call an (interactive) algorithm CEPT in the following, we mean that (if necessary) the inputs are generated by $\mathcal{I}$ (and $\mathcal{D}$ is applied to the output).

Suppose for simplicity that it halts with probability 1. Then the output of a normal rewinding strategy RWS is distributed like the real protocol output.[51] By normality, RWS is CEPT. By relative efficiency, $\mathrm{R}^{\mathcal{C}_0}(\mathsf{V}^*)$ is CEPT. Also, by assumption, the “reduction” $\mathrm{R}^{\mathcal{C}_0}(\mathsf{V}^*)$ behaves (as a system) exactly like RWS. By indistinguishability of $\mathcal{C}_0$ and $\mathcal{C}_1$, the standard reduction shows that $\mathrm{R}^{\mathcal{C}_1}(\mathsf{V}^*)$ is CEPT and the output of $\mathrm{R}^{\mathcal{C}_1}(\mathsf{V}^*)$ is (computationally) indistinguishable from $\mathrm{R}^{\mathcal{C}_0}(\mathsf{V}^*)$ (and hence the real protocol). By relative efficiency of Sim, $\mathsf{Sim}(\mathsf{V}^*)$ is CEPT (with environment $\mathcal{I}, \mathcal{D}$). Since $\mathrm{R}^{\mathcal{C}_1}(\mathsf{V}^*)$ is behaves (as a system) exactly as $\mathsf{Sim}(\mathsf{V}^*)$, the output of $\mathsf{Sim}(\mathsf{V}^*)$ and $\langle \mathsf{P}, \mathsf{V}^* \rangle$ is indistinguishable. Thus Sim handles CEPT adversaries in CEPT. □

By Lemma 5.6.23, all of our examples in Example 5.6.22 are not only secure against a priori PPT adversaries, but have CEPT simulation against *designated CEPT* adversaries.

Remark 5.6.24 (More precise runtime bounds). We saw for $\mathrm{G3C}_{\mathrm{GK}}$, that the runtime of the simulator Sim and the rewinding strategy RWS are *very closely related*. For this, we used “boxing” and “unboxing” (and that commitment computations only depend on message lengths). Such a close relation of runtime is typical, since in most security proofs only rewinding and bookkeeping introduces (significant) changes in the runtime. Hence, our extendability results are relatively *crude feasibility results*, assuring that zero-knowledge extends to CEPT adversaries.

[51] If $\langle \mathsf{P}, \mathsf{V}^* \rangle_{\mathcal{I}}$ halts with probability $1 - \nu$, then the output (including `nohalt`) has statistical distance at most $\mathrm{poly}_{\mathrm{virt}} \cdot \nu$. Since ν is bounded by the virtuality of $\langle \mathsf{P}, \mathsf{V}^* \rangle_{\mathcal{I}}$ anyway, the rest of the proof works without change.

5.6.4. Sequential Zero-Knowledge from Benign Simulation

To prove sequential that sequential zero-knowledge follows from auxiliary input zero-knowledge, we had to rely on the hybrid argument, which hides a lot of complexity. For benign simulation, it is easy to prove that it composes sequentially. Conceptually this follows from:

- Using that rewinding strategies "compose sequentially".
- Using that relative efficiency *with runtime tightness* "composes sequentially".
- Using that simple assumptions "compose sequentially", which is a very fancy way to say that we rely on "repeated trials".
- Hence, benign "composes sequentially".

Remark 5.6.25 (Lifting normality and relative efficiency). For brevity's sake, we do not explicitly lift rewinding strategies and relative efficiency to the sequential composition setting, i.e. we do not explicitly define what "composes sequentially" means in that setting. It is straightforward to define by using an (environmental) adversary and replacing access to the objects $\mathbb{O}_0$, $\mathbb{O}_1$ of interest (e.g. RWS and $\langle \mathsf{P}, \cdot \rangle$ for normality) by repeated access, i.e. $\mathsf{rep}(\mathbb{O}_0)$, $\mathsf{rep}(\mathbb{O}_1)$. We note that the tightness parameters are unaffected (since the notions were already "perfect").

Lemma 5.6.26 (Sequential zero-knowledge from benign simulation)**.** *Let* (P, V) *be an argument system. Suppose* Sim *is a benign simulator (for auxiliary input zero-knowledge). Then* (P, V) *is sequential zero-knowledge.*

Proof sketch. Let $(\mathcal{E}, \mathsf{V}^*)$ be the adversary trying to distinguish $\mathbb{O}_{\mathsf{P}}$ and $\mathbb{O}_{\mathsf{Sim}}$. Let RWS be the normal rewinding strategy of Sim. Let R be reduction and $\mathcal{C}_0$, $\mathcal{C}_1$ be the simple assumption.

Step 1 (Sequential composition of RWS**):** Let $\mathsf{poly}_{\mathsf{time}}$ and $\mathsf{poly}_{\mathsf{virt}}$ be the runtime and probability tightness of RWS. Let $\mathbb{O}_{\mathsf{P}} \mathrel{\widehat{=}} \mathsf{rep}(\langle \mathsf{P}, \mathsf{V}^* \rangle)$ and let $\mathbb{O}_{\mathsf{RWS}} \mathrel{\widehat{=}} \mathsf{rep}(\mathsf{RWS}^{\mathsf{V}^*})$. Suppose for simplicity that $(\mathcal{E}, \mathsf{V}^*)$ always halts. Then we know that for *any input*, the state of V^* after RWS is identically distributed to the state after interaction with P (by normality). Hence, replacing $\mathbb{O}_{\mathsf{P}}$ with $\mathbb{O}_{\mathsf{RWS}}$ only affects the runtime. Now, we lift Lemma 5.6.5 to the sequential setting.

Define $T_{\mathsf{RWS},i}$ resp. $T_{\mathsf{P},i}$ as the time spent in the i-th invocation of $\mathbb{O}_{\mathsf{RWS}}$ resp. $\mathbb{O}_{\mathsf{P}}$. Note that

$$\begin{aligned} \mathbb{E}[\mathsf{time}_{\mathsf{RWS}+\mathsf{V}^*}(\langle \mathcal{E}, \mathbb{O}_{\mathsf{RWS}} \rangle)] &= \sum_i \mathbb{E}[T_{\mathsf{RWS},i}] \\ &\leq \mathsf{poly}_{\mathsf{time}} \cdot \sum_i \mathbb{E}[T_{\mathsf{P},i}] \\ &= \mathsf{poly}_{\mathsf{time}} \cdot \mathbb{E}[\mathsf{time}_{\mathsf{P}+\mathsf{V}^*}(\langle \mathcal{E}, \mathbb{O}_{\mathsf{P}} \rangle)] \end{aligned}$$

where normality is applied for each i.

Suppose $(\mathcal{E}', \mathsf{V}')$ are `timeout`-modifications according to Lemma 5.3.12. By probability tightness, the probability that the i-th iteration of RWS runs into a `timeout` event is at most $\mathsf{poly}_{\mathsf{virt}}$-fold the probability for P to run into a `timeout` event. Consequently, the virtuality is increased by at most a factor of $\mathsf{poly}_{\mathsf{virt}}$.

All in all, we have shown that $\mathcal{O}_{\mathsf{RWS}}$ is a "sequential rewinding strategy" with runtime tightness $\mathsf{poly}_{\mathsf{time}}$, probability tightness $\mathsf{poly}_{\mathsf{virt}}$, and perfect output distribution;[52] and we lifted Lemma 5.6.5.

Step 2 (Relative efficiency composes sequentially): Let $\mathcal{O}_{\mathrm{R}^{\mathcal{C}_b}} \mathrel{\widehat{=}} \mathsf{rep}(\mathrm{R}^{\mathcal{C}_b}(\mathsf{V}^*))$ for $b \in \{0,1\}$, and let $\mathcal{O}_{\mathsf{Sim}} \mathrel{\widehat{=}} \mathsf{rep}(\mathsf{Sim}(\mathsf{V}^*))$. Suppose Sim is efficient relative to $\mathrm{R}^{\mathcal{C}_1}$ with runtime tightness $(\mathsf{poly}_{\mathsf{time}}, \mathsf{poly}_{\mathsf{virt}})$. Then the oracle $\mathcal{O}_{\mathsf{Sim}}$ is efficient relative to $\mathcal{O}_{\mathrm{R}^{\mathcal{C}_1}}$ with runtime tightness poly. Namely, for any $(\mathcal{E}, \mathsf{V}^*)$,

$$\begin{aligned}\mathbb{E}[\mathsf{time}_{\mathsf{Sim}+\mathsf{V}^*}(\langle \mathcal{E}, \mathcal{O}_{\mathsf{Sim}}\rangle)] &= \sum_i \mathbb{E}[T_{\mathsf{Sim},i}] \\ &\le \mathsf{poly}_{\mathsf{time}} \sum_i \mathbb{E}[T_{\mathrm{R}^{\mathcal{C}_1},i}] \\ &= \mathsf{poly}_{\mathsf{time}} \cdot \mathbb{E}[\mathsf{time}_{\mathrm{R}^{\mathcal{C}_1}}(\langle \mathcal{E}, \mathcal{O}_{\mathsf{P}}\rangle)]\end{aligned}$$

where $T_{\mathsf{Sim},i}$ resp. $T_{\mathrm{R}^{\mathcal{C}_1}}$ denotes the time for the i-th invocation of the respective oracle. This again follows by comparing i-th invocations, and using that output distributions are identical by assumption. And as for RWS, we can lift the runtime guarantees to the sequential setting, including virtualities. That is, if the virtually expected time is (t, ε) with $\mathcal{O}_{\mathrm{R}^{\mathcal{C}_1}}$, then it is $(\mathsf{poly}_{\mathsf{time}} \cdot t, \mathsf{poly}_{\mathsf{virt}} \cdot \varepsilon)$ with $\mathcal{O}_{\mathsf{Sim}}$. The same holds for $\mathcal{O}_{\mathsf{RWS}}$ and $\mathcal{O}_{\mathrm{R}^{\mathcal{C}_0}}$.

Step 3 (Indistinguishability of $\mathcal{O}_{\mathrm{R}^{\mathcal{C}_0}}$ and $\mathcal{O}_{\mathrm{R}^{\mathcal{C}_1}}$): It is obvious that indistinguishability of $\mathcal{O}_{\mathrm{R}^{\mathcal{C}_0}}$ and $\mathcal{O}_{\mathrm{R}^{\mathcal{C}_1}}$ reduces to indistinguishability of $\mathcal{C}_0$ and $\mathcal{C}_1$ under repeated trials. (Each invocation of $\mathcal{O}_{\mathrm{R}^{\mathcal{C}_0}}$ (resp. $\mathcal{O}_{\mathrm{R}^{\mathcal{C}_1}}$) is another trial.) By Corollary 5.6.17, simple assumptions are indistinguishable under repeated trials. (It is vital that $\mathrm{R}^{\mathcal{C}_0}$ is CEPT. That follows from Steps 1 and 2.)

Step 4 (Benign composes sequentially): From Steps 1 to 3, it follows immediately that benign "composes sequentially". More concretely, it follows that $(\mathcal{E}, \mathsf{V}^*)$ cannot distinguish $\mathcal{O}_{\mathsf{P}}$ and $\mathcal{O}_{\mathsf{Sim}}$, and in particular, an execution with $\mathcal{O}_{\mathsf{Sim}}$ is again CEPT. □

5.7. Sketched Application to SFE

We very briefly recall security definitions for SFE, but assume basic familiarity with the topic. Again, we adopt a uniform complexity setting with universal simulation. As with zero-knowledge, we therefore need an "environmental" adversary.

5.7.1. Definitions

Let n be a constant in λ and consider n interacting parties. In the setting of SFE, n parties wish to jointly compute a (probabilistic) functionality $\mathsf{f}\colon (\{0,1\}^*)^n \to (\{0,1\}^*)^n$ implemented by the algorithm f. We demand that f is a priori PPT in λ. The parties input $x_1, \dots, x_n$ and, at the end of the protocol, output $y_1, \dots, y_n$. A **protocol** π consists of algorithms $(\pi_1, \dots, \pi_n)$, such that each party i executes $\pi_i(x_i)$.[53] We assume the parties have *secure channels* for communication, i.e. an eavesdropping adversary only learns message lengths and communications proceeds in rounds.

[52] If the probability for non-halting executions is not 0, the easiest way is to argue by truncating after say 2^λ steps and using statistical closeness to $(\mathcal{E}, \mathsf{V}^*)$. But a closer inspection shows that any $\mathsf{poly}_{\mathsf{time}}$ is fine since $\infty \le \infty$. For $\mathsf{poly}_{\mathsf{virt}}$ a closer inspection shows that it does not change either. This is unsurprising since virtuality must at least remove non-halting executions anyway.

[53] Typically the π_i are a priori PPT, but as with zero-knowledge, we do not rely on this to specify and prove security.

We call a party **corrupted**, if it is controlled by the adversary. We restrict to **static corruption**, that is, for execution of a protocol π the subset $I \subseteq \{1, \dots, n\}$ of the corrupted parties is fixed from the start (and does not (adaptively) grow).

To shorten our exposition, we start with the **hybrid model**, and treat the real model as a special case. In the f-hybrid model, we assume (repeated) access to an ideal functionality f. A protocol π may use f, and we sometimes write π^f to emphasize this. Let π^f be a protocol implementing a functionality g in the f-hybrid model. Let $\mathcal{A}$ be an adversary corrupting the set $I \subseteq \{1, \dots, n\}$ of parties Let $\vec{x} = (x_1, \dots, x_n)$, $\vec{r} = (r_1, \dots, r_n, r_{\mathcal{A}}, r')$ denote the inputs and randomness of the parties. Here r' denotes the randomness used in ideal functionalities. By *aux* we denote the auxiliary input of $\mathcal{A}$. The adversary's input will be $\{x_i\}_{i \in I}$, *aux* and I. The computation of π proceeds in rounds. The parties can also query an instance of the functionality f. In the end, all parties return outputs y_i, and the adversary outputs $y_{\mathcal{A}}$; we write $\vec{y} = (y_1, \dots, y_n, y_{\mathcal{A}})$ for all outputs.

We denote by $\mathsf{Hybrid}^f_{\pi,\mathcal{A}}(\lambda, \vec{x}, aux, I; \vec{r})$ the output $\vec{y}$ of the execution of the protocol π^f where adversary $\mathcal{A}$ controls the parties in I and the inputs to all parties is $\vec{x}$ (and randomness $\vec{r}$). Since the parties have access to f, we call this the **f-hybrid model**.

In the **real model**, $\mathsf{Real}_{\pi,\mathcal{A}}(\lambda, \vec{x}, aux, I; \vec{r})$, denotes the output of a real execution. This is defined as in the hybrid model, except that there is no hybrid functionality (i.e. f is the null-functionality).

We denote by $\mathsf{Ideal}_{g,\mathsf{Sim}}(\lambda, \{x_i\}_{i \in I}, \mathrm{code}(\mathcal{A}), aux, I; \vec{r})$ the output $\vec{y} = (y_1, \dots, y_n, y_{\mathsf{Sim}})$ of an execution in the **ideal model** with functionality g and ideal adversary Sim, called (universal) simulator. Here, the honest parties hand their inputs to g and output what they receive from g; inputs for corrupted parties may be provided by Sim. The simulator is given $\{x_i\}_{i \in I}$, *aux*, I, and $\mathrm{code}(\mathcal{A})$ as input. (As usual, we often omit $\mathrm{code}(\mathcal{A})$ when it is clear from the context; it is only required for universal simulation.[54])

We extend the definition of Hybrid, Real and Ideal to adaptive sequential composition, where an "environment" $\mathcal{E}$ provides inputs to executions of Hybrid, Real or Ideal (which choose fresh randomness). We denote this by $\mathsf{Hybrid}^f_{\pi,\mathcal{A}}(\lambda, \mathcal{E})$, or $\mathsf{Real}_{\pi,\mathcal{A}}(\lambda, \mathcal{E})$, or $\mathsf{Ideal}_{g,\mathsf{Sim}}(\lambda, \mathcal{E})$. After each execution, $\mathcal{E}$ learns all outputs, and can (adaptively) choose inputs for further executions. More formally, $\mathcal{E}$ is given access to either $\mathcal{O}_{\pi,\mathcal{A}}$ or $\mathcal{O}_{g,\mathsf{Sim}}$, which take as inputs $(\vec{x}, code, aux, I)$ and output $\vec{y}$ (which includes $y_{\mathcal{A}}$), i.e. $\mathcal{O}_{\pi,\mathcal{A}}(\vec{x}, aux, I) = \mathsf{Real}_{\pi,\mathcal{A}}(\vec{x}, aux, I)$ and $\mathcal{O}_{g,\mathsf{Sim}}(\vec{x}, aux, I) = \mathsf{Ideal}_{g,\mathsf{Sim},I}(\vec{x}, aux, I)$. (We omitted *code*, since it is always $\mathrm{code}(\mathcal{A})$.) Note that $\mathcal{E}$ can adaptively choose the set I of corrupted parties as well.

Definition 5.7.1. Let π be a protocol for g in the f-hybrid model. We say an adversary $(\mathcal{E}, \mathcal{A})$ is $\mathcal{T}$-time (e.g. CEPT), if $\mathrm{time}(\mathsf{Hybrid}^f_{\pi,\mathcal{A}}(\lambda, \mathcal{E}))$ is $\mathcal{T}$-time, where $(\vec{y}, aux, state) \leftarrow \mathcal{G}$. (We stress that the time to compute f is included here.)

The protocol π is said to **sequentially t-securely** compute g against $\mathcal{T}$-time adversaries, if there exists a (universal) ideal adversary Sim, for any $\mathcal{T}$-time adversary $(\mathcal{E}, \mathcal{A})$, where $\mathcal{E}$ only corrupts subset of size at most t, we have

- $\mathsf{Real}^f_{\pi,\mathcal{A}}(\lambda, \mathcal{E}) \overset{c}{\approx} \mathsf{Ideal}_{g,\mathsf{Sim}}(\lambda, \mathcal{E})$;
- $\mathrm{time}(\mathsf{Ideal}_{g,\mathsf{Sim}}(\lambda, \mathcal{E}))$ is $\mathcal{T}$-time (e.g. CEPT);

Auxiliary input t-security is defined by restricting $\mathcal{E}$ to one query only; in this case, we usually write $\mathcal{G}$ instead of $\mathcal{E}$.

Lemma 5.7.2. *Auxiliary input t-security and sequential t-security are equivalent.*

[54] In our setting, universal and existential simulation coincide, just like for zero-knowledge.

Proof sketch. This is a straightforward application of the hybrid lemma. □

Note that there is no hybrid functionality f in the ideal world. The simulator must provide the hybrid functionalities. In particular, it is essential that f itself is efficient. Also note that we require that Sim is universal in both $\mathcal{A}$ and I. This does not strengthen the security in our setting, but simplifies discussions.[55] Some more remarks are in order.

Remark 5.7.3. We note that neither our results nor our definitions require black-box simulation. This is unlike [KL08]. However, the overhead of non-black-box simulation seems to preclude its use — at least the technique of [Bar01].

Remark 5.7.4. The notion of benign simulation can be extended to simulators for SFE.

Remark 5.7.5 (Zero-knowledge as an ideal functionality). The ideal zero-knowledge functionality takes as input (x, w) and outputs x to the verifier if $(w, x) \in \mathcal{R}$, else $\perp$. The protocol $\mathrm{G3C}_{\mathrm{GK}}$ is not (proven) t-secure as an ideal zero-knowledge functionality, because it does not seem to be a proof of knowledge, i.e. the witness cannot be extracted. Lindell [Lin13] describes a 5-move protocol, which is a zero-knowledge proof of knowledge, hence realizes the ideal zero-knowledge functionality. Its simulator handles CEPT adversaries in CEPT. (The simulation is benign.)

Remark 5.7.6 (Proofs of knowledge). Communication efficient (zero-knowledge) proofs *of knowledge* often have a superlinear overhead for extraction in the witness size. This can break compatibility with CEPT completely, for example if extraction has a quadratic overhead in the witness size, then fat-tailed input distributions lead to inefficient extraction. Again, the problem are *expected* size inputs, and can be mitigated to some extent by size-guarding.

5.7.2. Modular Sequential Composition

In the following, we denote substituting an (ideal) subprotocol f by a (real) subprotocol ρ in π^f as π^ρ. Similarly we write $\pi^{f_1,\dots,f_m}$ resp. $\pi^{\rho_1,\dots,\rho_m}$ for substituting in multiple protocols. We need to assume that π proceeds in rounds, making only one subprotocol call per round. Moreover, all (honest) parties always call the same subprotocol.[56]

We can now state our adaption of [KL08, Theorem 12]. Unlike [KL08, Theorem 12], we assume the protocols are secure against CEPT adversaries (in our sense).

Theorem 5.7.7. *Let $f_1, \dots, f_m$ and g be ideal n-party functionalities. Let π be an n-party protocol that t-securely computes g against CEPT adversaries in the $(f_1, \dots, f_m)$-hybrid model. Suppose that π makes no more than one call to an ideal functionality in each round, that is, the functionalities f_i are used strictly sequentially. Let $\rho_1, \dots, \rho_m$ be n-party protocols so that ρ_i t-securely compute f_i against CEPT adversaries (in the real model). Then $\pi^{\rho_1,\dots,\rho_m}$ t-securely computes g against CEPT adversaries (in the real model).*

The proof of Theorem 5.7.7 is straightforward — it is essentially as in [Can00]. That is:

[55] We require constant n, so separate simulators (for the constantly many $I \subseteq \{1, \dots, n\}$) can be merged into one. Due to the a posteriori efficiency setting, existential simulators are universal anyway, as for zero-knowledge.

[56] See [Can00] for details the restrictions imposed on π.

1. We construct from $(\mathcal{G}, \mathcal{A})$ the obvious adversary $(\mathcal{E}_\rho, \mathcal{A}_\rho)$ against sequential t-security of ρ. By assumption, we can replace $\mathcal{A}_\rho$ with Sim_ρ (and ρ with f). The execution remains efficient and the output is indistinguishable, i.e.

$$\mathsf{Real}_{\rho,\mathcal{A}_\rho}(\lambda, \mathcal{E}_\rho) \overset{c}{\approx} \mathsf{Ideal}^{f}_{\rho,\mathsf{Sim}_\rho}(\lambda, \mathcal{E}_\rho)$$

   Thus, π^ρ was effectively replaced by π^f.

2. Now, we construct an adversary $(\mathcal{G}, \mathcal{A}_\pi)$ against t-security of π in the f-hybrid model. By assumption, we can replace $\mathcal{A}_\pi$ with Sim_π (and π with g). Again, the execution remains efficient and the output is indistinguishable, i.e.

$$\mathsf{Hybrid}^{f}_{\pi,\mathcal{A}_\pi}(\lambda, \mathcal{G}) \overset{c}{\approx} \mathsf{Ideal}_{g,\mathsf{Sim}_\pi}(\lambda, \mathcal{G})$$

   This concludes the proof.

Note that we can avoid the requirement of [KL08], that the simulator for ρ is efficient *in any interaction*. Weak relative efficiency of simulation is sufficient. The hybrid lemma takes care of the all the hairy details.[57]

Now, we sketch the proof in more detail. For brevity's sake, we will not repeat that t-security *against CEPT adversaries* is considered in every statement of "X securely computes Y". We note that, by assumption on $\pi^{f_1,\dots,f_m}$, the real protocol $\pi^{\rho_1,\dots,\rho_m}$ does not interleave executions. That is, only one instance of a subprotocol ρ_i is executed at a time.

Proof sketch. First, we simplify the situation by considering only one hybrid functionality f and protocol ρ. Since m is a constant, it will be evident that the proof easily extends, e.g. by going through m hybrid models.

Before we continue, we clarify and revert an important notational difference: We used the notation $\mathsf{rep}(\mathcal{O})$ for *repeated* access to independent instance of $\mathcal{O}$. In SFE/MPC, it is common that π^ρ resp. π^g denotes that π has *repeated* access to independent instances of ρ resp. g.

As noted before Definition 5.7.1, we view security as oracle indistinguishability. As with zero-knowledge, the hybrid lemma shows that $\mathsf{rep}(\mathcal{O}_{\rho,\mathcal{A}_\rho})$ is weakly efficient relative to $\mathsf{rep}(\mathcal{O}_{f,\mathsf{Sim}_\rho})$ and $\mathsf{rep}(\mathcal{O}_{\rho,\mathcal{A}_\rho}) \overset{c}{\approx} \mathsf{rep}(\mathcal{O}_{f,\mathsf{Sim}_\rho})$. That is, the analogue of sequential t-security (for a single protocol ρ) holds.

We argue in games. **Game** G_0 is the real execution π^ρ, that is $\mathsf{Real}_{\pi,\mathcal{A}}(\lambda, \mathcal{G})$.

In **Game** G_1, we prepare to replace all instances ρ by f. For this, we interpret the calling "environment" of ρ as $\mathcal{E}_\rho$. That is, $\mathcal{E}_\rho$ executes π using access to $\mathsf{rep}(\mathcal{O}_{\rho,\mathcal{A}_\rho})$. The adversary $\mathcal{A}$ is now split and executed partially by $\mathcal{E}_\rho$ and $\mathcal{A}_\rho$. Here $\mathcal{E}_\rho$ simulates the everything (the game, honest parties and $\mathcal{A}$) outside the subprotocol calls to ρ, whereas $\mathcal{A}_\rho$ emulates $\mathcal{A}$ (only) the subprotocol calls (and receives the state of $\mathcal{A}$ via *aux*). Here, it is essential that the calls are sequential. The changes from G_0 to G_1 are conceptual. Everything is efficient if (and only if) it was efficient before and the output is identically distributed.

In **Game** G_2, we replace $\mathcal{O}_{\rho,\mathcal{A}_\rho}$ by $\mathcal{O}_{f,\mathsf{Sim}_\rho}$. As noted before, the hybrid lemma implies that

$$\mathrm{G}_1 \equiv \mathsf{Real}_{\rho,\mathcal{A}_\rho}(\mathcal{E}_\rho) \equiv \mathcal{E}_\rho^{\mathsf{rep}(\mathcal{O}_{\rho,\mathcal{A}_\rho})} \overset{c}{\approx} \mathcal{E}_\rho^{\mathsf{rep}(\mathcal{O}_{f,\mathsf{Sim}_\rho})} \equiv \mathsf{Ideal}_{f,\mathsf{Sim}_\rho}(\mathcal{E}_\rho) \equiv \mathrm{G}_2$$

[57] It is unsurprising that the proof of the modular sequential composition theorem in [KL08], is more complex, since it effectively *is* the hybrid argument.

and both executions are efficient. Thus, we have substituted ρ by f in π. Now, we are effectively in the f-hybrid model.

Game G_3 undoes the changes of G_1, i.e. we revert to $\mathcal{E} \mathrel{\widehat{=}} \mathcal{G}$ and π, but now, we have $\pi^{\mathsf{rep}(f)}$ instead of $\pi^{\mathsf{rep}(\rho)}$. We call the resulting “network” adversary $\mathcal{A}_\pi$. This change is conceptual and does not affect output or efficiency.

In **Game** G_4, we replace $\pi^{\mathsf{rep}(f)}$ (resp. $\mathcal{A}_\pi$) by g (resp. Sim_π). Since π t-securely computes g in the f-hybrid model, we find

$$G_3 \equiv \mathsf{Hybrid}^{f}_{\pi,\mathcal{A}_\pi}(\mathcal{G}) \equiv \mathcal{E}_\pi^{\mathcal{O}_{\pi,\mathcal{A}_\pi}} \overset{c}{\approx} \mathcal{E}_\pi^{\mathcal{O}_{g,\mathsf{Sim}_\pi}} \equiv \mathsf{Ideal}_{g,\mathsf{Sim}_\pi}(\mathcal{E}_\pi) \equiv G_4$$

and G_4 is efficient if G_4 is. Thus $\mathsf{Ideal}_{g,\mathsf{Sim}_\pi}(\mathcal{E}_\pi)$ CEPT.

The construction of the simulator Sim for π^ρ follows from the above. That is, Sim runs Sim_ρ for $\mathcal{A}_\rho$ in subprotocol calls to ρ and Sim_π for π. □

5.8. Conclusion and Open Problems

At the example of zero-knowledge and a sketched application to SFE, we demonstrated that the notion of computationally expected polynomial time is a useful and viable alternative to EPT. We also gave a “philosophical” motivation why EPT should be enlarged to CEPT, namely distinguishing-closedness. However, we leave open many minor and major questions and directions.

Beyond Negligible Advantage. The most important question may well be the *(in)compatibility of CPPT/CEPT and superpolynomial hardness assumptions.* Concretely, consider a one-way function where we assume that no PPT adversary can invert with probability better than $\mathcal{O}(2^{-\lambda/2})$. W.r.t. CPPT/CEPT, such assumptions cannot exist, since with probability $\mathcal{O}(2^{-\lambda/4})$, a CPPT/CEPT adversary may brute-force a preimage.

It is a critical question, whether this is a fundamental problem, or just another technical artefact. If CPPT/CEPT is incompatible with subexponential hardness assumptions, then protocols which rely on such are very likely incompatible with CPPT

Quantifiability, Tightness, Constructivity. For a more quantifiable notion of security, we need to better tackle the question of *tightness* of reductions, simulations, etc. The interpretation and treatment of the virtuality error for a good notion of tightness is non-trivial. Moreover, constructivity of security reductions is an interesting and important question, as we used the existence of (in general not computable) polynomial bounds in many places. In Appendices C.5.3.1 and C.5.3.2, we explore these questions very briefly.

Efficiency Artefacts. In several situations, expected polynomial size inputs and messages resulted in rather strong requirements and fickle behaviour. Size-guards (Appendix C.5.4.3) are a mitigation. A more natural alternative is to investigate the efficiency class of *expected time strict space (EPT/SPS)* machines.

More Abstract Questions. Our “general” treatment of runtime provides the central results only for algebra-tailed runtime classes. Indeed, we even lack a definition of well-behaved runtime classes, for which we can expect such results to hold. Such a definition and extensions, as well as incorporating different advantage classes, are open. This may also lead to insights regarding superpolynomial hardness and CEPT, or vice versa.

6. Conclusion

We conclude with an outlook on open questions and interesting research directions.

Rational Representatives. In Chapter 3 we presented our short relaxed range proof Sharp. The relaxed soundness definition and its efficiency crucially relies on the interpretation of a range through rational representatives. We also noted that rational representatives are not as well-behaved as the usual integer representatives, making them hard(er) to use. This raises the question in which other settings rational representatives might prove useful. Moreover, the proof of our core lemma does not mirror the intuitive nature of its statement. We would be delighted to see a simpler, stronger and more generally applicable proof.

Better Handling of Class Groups. When relying on class groups of imaginary quadratic orders in Chapter 3, we either need to assume a trusted setup, or we need strengthened hardness assumptions in the class group to it secure with transparent setup. The core problem and open question is, whether it is possible to efficiently circumvent this without increasing the proof size.[1] Moreover, we leave open the question of whether it is possible to relate our assumptions in groups of hidden order (cf. Lemma A.1.7), or how to abstractly model this setting, where sampling might leak something, e.g. in an appropriate generic or algebraic group model. An efficient reverse sampling algorithm in class groups would resolve all of these questions (in class groups), but its existence appears to be an open problem.

Short-Circuit Extraction. In Chapter 4 we introduce short-circuit extraction to prove better bounds on runtime tightness of extraction, aiming for better bounds on concrete security. In Appendix B.9, we present two candidate extractors for which a formal runtime analysis is an open question. If the candidate with optimal knowledge error, would have runtime bounded by the worst-case size of the tree of challenges which must be found, then this would provide the tightest reduction (without knowledge assumptions) for Bulletproofs and $\mathrm{QESA}_{\mathrm{ZK}}$ by a large margin. Another question is if and how short-circuit extraction can be handled more generally, and how generic (sequential) composition results (e.g. analogous to those recalled in Section 2.5.2) would look like.

Computationally Expected Polynomial Time. In Section 5.8, we already discuss several open questions and directions for CEPT. We only recall the main questions here: Firstly, due to the definition of CEPT it is not immediately clear what a good notion of tightness would be in this setting. In that regard, it may be interesting to investigate a non-asymptotic variant of CEPT. Secondly, again due to the definition of CEPT, certain hardness assumptions cannot hold against CEPT adversaries, which raises the question if this breaks security reductions relying on such assumptions (or if the reduction can sidestep the problem), and also whether this is a fundamental problem with CEPT and analogous definitions, or if small changes to the definition are sufficient to fix it.

[1] We show in [CKLR21a] that it is possible to rely on ElGamal commitments, but this significantly increases the proof size.

Bibliography

[AC20] Thomas Attema and Ronald Cramer. “Compressed Σ-Protocol Theory and Practical Application to Plug & Play Secure Algorithmics”. In: *Advances in Cryptology – CRYPTO 2020, Part III.* Ed. by Daniele Micciancio and Thomas Ristenpart. Vol. 12172. Lecture Notes in Computer Science. Springer, Heidelberg, Aug. 2020, pp. 513–543. DOI: `10.1007/978-3-030-56877-1_18`.

[ACK21] Thomas Attema, Ronald Cramer, and Lisa Kohl. “A Compressed Σ-Protocol Theory for Lattices”. In: *Advances in Cryptology – CRYPTO 2021, Part II.* Ed. by Tal Malkin and Chris Peikert. Vol. 12826. Lecture Notes in Computer Science. Virtual Event: Springer, Heidelberg, Aug. 2021, pp. 549–579. DOI: `10.1007/978-3-030-84245-1_19`.

[ACR21] Thomas Attema, Ronald Cramer, and Matthieu Rambaud. “Compressed Σ-Protocols for Bilinear Group Arithmetic Circuits and Application to Logarithmic Transparent Threshold Signatures”. In: *Advances in Cryptology – ASIACRYPT 2021, Part IV.* Ed. by Mehdi Tibouchi and Huaxiong Wang. Vol. 13093. Lecture Notes in Computer Science. Springer, Heidelberg, Dec. 2021, pp. 526–556. DOI: `10.1007/978-3-030-92068-5_18`.

[ADOS22] Damiano Abram, Ivan Damgård, Claudio Orlandi, and Peter Scholl. “An Algebraic Framework for Silent Preprocessing with Trustless Setup and Active Security”. In: *Advances in Cryptology – CRYPTO 2022.* Ed. by Yevgeniy Dodis and Thomas Shrimpton. Springer Nature Switzerland, 2022.

[AFK21] Thomas Attema, Serge Fehr, and Michael Klooß. *Fiat-Shamir Transformation of Multi-Round Interactive Proofs.* Cryptology ePrint Archive, Report 2021/1377. `https://eprint.iacr.org/2021/1377`. 2021.

[AFK22] Thomas Attema, Serge Fehr, and Michael Klooß. “Fiat-Shamir Transformation of Multi-Round Interactive Proofs”. In: *Theory of Cryptography - 20th International Conference, TCC 2022, Chicage, IL, USA, November 7-10, 2022, Proceedings, Part ?* Ed. by Eike Kiltz and Vinod Vaikuntanathan. Lecture Notes in Computer Science. To appear. Springer, 2022.

[AG] D. F. Aranha and C. P. L. Gouvêa. *RELIC is an Efficient LIbrary for Cryptography.* `https://github.com/relic-toolkit/relic`.

[AHIV17] Scott Ames, Carmit Hazay, Yuval Ishai, and Muthuramakrishnan Venkitasubramaniam. “Ligero: Lightweight Sublinear Arguments Without a Trusted Setup”. In: *ACM CCS 2017: 24th Conference on Computer and Communications Security.* Ed. by Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu. ACM Press, 2017, pp. 2087–2104. DOI: `10.1145/3133956.3134104`.

[AHK20] Thomas Agrikola, Dennis Hofheinz, and Julia Kastner. “On Instantiating the Algebraic Group Model from Falsifiable Assumptions”. In: *Advances in Cryptology – EUROCRYPT 2020, Part II.* Ed. by Anne Canteaut and Yuval Ishai. Vol. 12106. Lecture Notes in Computer Science. Springer, Heidelberg, May 2020, pp. 96–126. DOI: `10.1007/978-3-030-45724-2_4`.

[AL21] Martin R. Albrecht and Russell W. F. Lai. “Subtractive Sets over Cyclotomic Rings - Limits of Schnorr-Like Arguments over Lattices”. In: *Advances in Cryptology – CRYPTO 2021, Part II*. Ed. by Tal Malkin and Chris Peikert. Vol. 12826. Lecture Notes in Computer Science. Virtual Event: Springer, Heidelberg, Aug. 2021, pp. 519–548. DOI: `10.1007/978-3-030-84245-1_18`.

[AVY] Oleg Andreev, Henry de Valence, and Cathie Yun. *dalek-cryptography bulletproofs*. `https://github.com/dalek-cryptography/bulletproofs`.

[Bar01] Boaz Barak. “How to Go Beyond the Black-Box Simulation Barrier”. In: *42nd Annual Symposium on Foundations of Computer Science*. IEEE Computer Society Press, Oct. 2001, pp. 106–115. DOI: `10.1109/SFCS.2001.959885`.

[BAZB20] Benedikt Bünz, Shashank Agrawal, Mahdi Zamani, and Dan Boneh. “Zether: Towards Privacy in a Smart Contract World”. In: *FC 2020: 24th International Conference on Financial Cryptography and Data Security*. Ed. by Joseph Bonneau and Nadia Heninger. Vol. 12059. Lecture Notes in Computer Science. Springer, Heidelberg, Feb. 2020, pp. 423–443. DOI: `10.1007/978-3-030-51280-4_23`.

[BBB+18] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. “Bulletproofs: Short Proofs for Confidential Transactions and More”. In: *2018 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2018, pp. 315–334. DOI: `10.1109/SP.2018.00020`.

[BBC+18] Carsten Baum, Jonathan Bootle, Andrea Cerulli, Rafaël del Pino, Jens Groth, and Vadim Lyubashevsky. “Sub-linear Lattice-Based Zero-Knowledge Arguments for Arithmetic Circuits”. In: *Advances in Cryptology – CRYPTO 2018, Part II*. Ed. by Hovav Shacham and Alexandra Boldyreva. Vol. 10992. Lecture Notes in Computer Science. Springer, Heidelberg, Aug. 2018, pp. 669–699. DOI: `10.1007/978-3-319-96881-0_23`.

[BBDE19] Johannes Blömer, Jan Bobolz, Denis Diemert, and Fabian Eidens. “Updatable Anonymous Credentials and Applications to Incentive Systems”. In: *ACM CCS 2019: 26th Conference on Computer and Communications Security*. Ed. by Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz. ACM Press, Nov. 2019, pp. 1671–1685. DOI: `10.1145/3319535.3354223`.

[BCC+16] Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Jens Groth, and Christophe Petit. “Efficient Zero-Knowledge Arguments for Arithmetic Circuits in the Discrete Log Setting”. In: *Advances in Cryptology – EUROCRYPT 2016, Part II*. Ed. by Marc Fischlin and Jean-Sébastien Coron. Vol. 9666. Lecture Notes in Computer Science. Springer, Heidelberg, May 2016, pp. 327–357. DOI: `10.1007/978-3-662-49896-5_12`.

[BCC88] Gilles Brassard, David Chaum, and Claude Crépeau. “Minimum Disclosure Proofs of Knowledge”. In: *J. Comput. Syst. Sci.* 37.2 (1988), pp. 156–189. DOI: `10.1016/0022-0000(88)90005-0`. URL: `https://doi.org/10.1016/0022-0000(88)90005-0`.

[BCFK15] Foteini Baldimtsi, Melissa Chase, Georg Fuchsbauer, and Markulf Kohlweiss. “Anonymous Transferable E-Cash”. In: *PKC 2015: 18th International Conference on Theory and Practice of Public Key Cryptography*. Ed. by Jonathan Katz. Vol. 9020. Lecture Notes in Computer Science. Springer, Heidelberg, 2015, pp. 101–124. DOI: `10.1007/978-3-662-46447-2_5`.

[BCG+13] Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, Eran Tromer, and Madars Virza. "SNARKs for C: Verifying Program Executions Succinctly and in Zero Knowledge". In: *Advances in Cryptology – CRYPTO 2013, Part II*. Ed. by Ran Canetti and Juan A. Garay. Vol. 8043. Lecture Notes in Computer Science. Springer, Heidelberg, Aug. 2013, pp. 90–108. DOI: 10.1007/978-3-642-40084-1_6.

[BCG+17] Jonathan Bootle, Andrea Cerulli, Essam Ghadafi, Jens Groth, Mohammad Hajiabadi, and Sune K. Jakobsen. "Linear-Time Zero-Knowledge Proofs for Arithmetic Circuit Satisfiability". In: *Advances in Cryptology – ASIACRYPT 2017, Part III*. Ed. by Tsuyoshi Takagi and Thomas Peyrin. Vol. 10626. Lecture Notes in Computer Science. Springer, Heidelberg, Dec. 2017, pp. 336–365. DOI: 10.1007/978-3-319-70700-6_12.

[BCPR16] Nir Bitansky, Ran Canetti, Omer Paneth, and Alon Rosen. "On the Existence of Extractable One-Way Functions". In: *SIAM J. Comput.* 45.5 (2016), pp. 1910–1952. DOI: 10.1137/140975048. URL: https://doi.org/10.1137/140975048.

[BCS16] Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. "Interactive Oracle Proofs". In: *TCC 2016-B: 14th Theory of Cryptography Conference, Part II*. Ed. by Martin Hirt and Adam D. Smith. Vol. 9986. Lecture Notes in Computer Science. Springer, Heidelberg, 2016, pp. 31–60. DOI: 10.1007/978-3-662-53644-5_2.

[BCS21] Jonathan Bootle, Alessandro Chiesa, and Katerina Sotiraki. "Sumcheck Arguments and Their Applications". In: *Advances in Cryptology – CRYPTO 2021, Part I*. Ed. by Tal Malkin and Chris Peikert. Vol. 12825. Lecture Notes in Computer Science. Virtual Event: Springer, Heidelberg, Aug. 2021, pp. 742–773. DOI: 10.1007/978-3-030-84242-0_26.

[BDFG21] Dan Boneh, Justin Drake, Ben Fisch, and Ariel Gabizon. "Halo Infinite: Proof-Carrying Data from Additive Polynomial Commitments". In: *Advances in Cryptology – CRYPTO 2021, Part I*. Ed. by Tal Malkin and Chris Peikert. Vol. 12825. Lecture Notes in Computer Science. Virtual Event: Springer, Heidelberg, Aug. 2021, pp. 649–680. DOI: 10.1007/978-3-030-84242-0_23.

[BDGJ20] Colin Boyd, Gareth T. Davies, Kristian Gjøsteen, and Yao Jiang. "Fast and Secure Updatable Encryption". In: *Advances in Cryptology – CRYPTO 2020, Part I*. Ed. by Daniele Micciancio and Thomas Ristenpart. Vol. 12170. Lecture Notes in Computer Science. Springer, Heidelberg, Aug. 2020, pp. 464–493. DOI: 10.1007/978-3-030-56784-2_16.

[BDL+18] Carsten Baum, Ivan Damgård, Vadim Lyubashevsky, Sabine Oechsner, and Chris Peikert. "More Efficient Commitments from Structured Lattice Assumptions". In: *SCN 18: 11th International Conference on Security in Communication Networks*. Ed. by Dario Catalano and Roberto De Prisco. Vol. 11035. Lecture Notes in Computer Science. Springer, Heidelberg, Sept. 2018, pp. 368–385. DOI: 10.1007/978-3-319-98113-0_20.

[BEB+] Bank of Canada, European Central Bank, Bank of Japan, Sveriges Riksbank, Swiss National Bank, Bank of England, Board of Governors Federal Reserve System, and Bank for International Settlements. *Central bank digital currencies: foundational principles and core features*. https://www.bis.org/publ/othp33.pdf.

[BEK+20] Jan Bobolz, Fabian Eidens, Stephan Krenn, Daniel Slamanig, and Christoph Striecks. "Privacy-Preserving Incentive Systems with Highly Efficient Point-Collection". In: *ASIACCS 20: 15th ACM Symposium on Information, Computer and Communications Security*. Ed. by Hung-Min Sun, Shiuh-Pyng Shieh, Guofei Gu, and Giuseppe Ateniese. ACM Press, Oct. 2020, pp. 319–333. DOI: 10.1145/3320269.3384769.

[BEKS20] Dan Boneh, Saba Eskandarian, Sam Kim, and Maurice Shih. "Improving Speed and Security in Updatable Encryption Schemes". In: *Advances in Cryptology – ASIACRYPT 2020, Part III.* Ed. by Shiho Moriai and Huaxiong Wang. Vol. 12493. Lecture Notes in Computer Science. Springer, Heidelberg, Dec. 2020, pp. 559–589. DOI: 10.1007/978-3-030-64840-4_19.

[Bel02] Mihir Bellare. "A Note on Negligible Functions". In: *Journal of Cryptology* 15.4 (Sept. 2002), pp. 271–284. DOI: 10.1007/s00145-002-0116-x.

[BFS20] Benedikt Bünz, Ben Fisch, and Alan Szepieniec. "Transparent SNARKs from DARK Compilers". In: *Advances in Cryptology – EUROCRYPT 2020, Part I.* Ed. by Anne Canteaut and Yuval Ishai. Vol. 12105. Lecture Notes in Computer Science. Springer, Heidelberg, May 2020, pp. 677–706. DOI: 10.1007/978-3-030-45721-1_24.

[BG10] Zvika Brakerski and Shafi Goldwasser. "Circular and Leakage Resilient Public-Key Encryption under Subgroup Indistinguishability - (or: Quadratic Residuosity Strikes Back)". In: *Advances in Cryptology – CRYPTO 2010.* Ed. by Tal Rabin. Vol. 6223. Lecture Notes in Computer Science. Springer, Heidelberg, Aug. 2010, pp. 1–20. DOI: 10.1007/978-3-642-14623-7_1.

[BG11] Mihir Bellare and Oded Goldreich. "On Probabilistic versus Deterministic Provers in the Definition of Proofs of Knowledge". In: *Studies in Complexity and Cryptography.* Vol. 6650. Lecture Notes in Computer Science. Springer, 2011, pp. 114–123.

[BG12] Stephanie Bayer and Jens Groth. "Efficient Zero-Knowledge Argument for Correctness of a Shuffle". In: *Advances in Cryptology – EUROCRYPT 2012.* Ed. by David Pointcheval and Thomas Johansson. Vol. 7237. Lecture Notes in Computer Science. Springer, Heidelberg, Apr. 2012, pp. 263–280. DOI: 10.1007/978-3-642-29011-4_17.

[BG18] Jonathan Bootle and Jens Groth. "Efficient Batch Zero-Knowledge Arguments for Low Degree Polynomials". In: *PKC 2018: 21st International Conference on Theory and Practice of Public Key Cryptography, Part II.* Ed. by Michel Abdalla and Ricardo Dahab. Vol. 10770. Lecture Notes in Computer Science. Springer, Heidelberg, Mar. 2018, pp. 561–588. DOI: 10.1007/978-3-319-76581-5_19.

[BG93] Mihir Bellare and Oded Goldreich. "On Defining Proofs of Knowledge". In: *Advances in Cryptology – CRYPTO'92.* Ed. by Ernest F. Brickell. Vol. 740. Lecture Notes in Computer Science. Springer, Heidelberg, Aug. 1993, pp. 390–420. DOI: 10.1007/3-540-48071-4_28.

[BGM17] Sean Bowe, Ariel Gabizon, and Ian Miers. *Scalable Multi-party Computation for zk-SNARK Parameters in the Random Beacon Model.* Cryptology ePrint Archive, Report 2017/1050. https://eprint.iacr.org/2017/1050. 2017.

[BJS10] Jean-François Biasse, Michael J. Jacobson, and Alan K. Silvester. "Security Estimates for Quadratic Field Based Cryptosystems". In: *ACISP 10: 15th Australasian Conference on Information Security and Privacy.* Ed. by Ron Steinfeld and Philip Hawkes. Vol. 6168. Lecture Notes in Computer Science. Springer, Heidelberg, July 2010, pp. 233–247.

[BKLP15] Fabrice Benhamouda, Stephan Krenn, Vadim Lyubashevsky, and Krzysztof Pietrzak. "Efficient Zero-Knowledge Proofs for Commitments from Learning with Errors over Rings". In: *ESORICS 2015: 20th European Symposium on Research in Computer Security, Part I.* Ed. by Günther Pernul, Peter Y. A. Ryan, and Edgar R. Weippl. Vol. 9326. Lecture Notes in Computer Science. Springer, Heidelberg, Sept. 2015, pp. 305–325. DOI: 10.1007/978-3-319-24174-6_16.

[BL04] Boaz Barak and Yehuda Lindell. "Strict Polynomial-Time in Simulation and Extraction". In: *SIAM J. Comput.* 33.4 (2004), pp. 738–818. DOI: 10.1137/S0097539703427975. URL: https://doi.org/10.1137/S0097539703427975.

[BL13] Foteini Baldimtsi and Anna Lysyanskaya. "Anonymous credentials light". In: *ACM CCS 2013: 20th Conference on Computer and Communications Security.* Ed. by Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung. ACM Press, Nov. 2013, pp. 1087–1098. DOI: 10.1145/2508859.2516687.

[BLNS20] Jonathan Bootle, Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Gregor Seiler. "A Non-PCP Approach to Succinct Quantum-Safe Zero-Knowledge". In: *Advances in Cryptology – CRYPTO 2020, Part II.* Ed. by Daniele Micciancio and Thomas Ristenpart. Vol. 12171. Lecture Notes in Computer Science. Springer, Heidelberg, Aug. 2020, pp. 441–469. DOI: 10.1007/978-3-030-56880-1_16.

[Blu81] Manuel Blum. "Coin Flipping by Telephone". In: *Advances in Cryptology – CRYPTO'81.* Ed. by Allen Gersho. Vol. ECE Report 82-04. U.C. Santa Barbara, Dept. of Elec. and Computer Eng., 1981, pp. 11–15.

[Blu86] Manuel Blum. "How to prove a theorem so no one else can claim it". In: *Proceedings of the International Congress of Mathematicians.* Vol. 1. 1986, p. 2.

[BMPR21] Christian Badertscher, Ueli Maurer, Christopher Portmann, and Guilherme Rito. "Revisiting (R)CCA Security and Replay Protection". In: *PKC 2021: 24th International Conference on Theory and Practice of Public Key Cryptography, Part II.* Ed. by Juan Garay. Vol. 12711. Lecture Notes in Computer Science. Springer, Heidelberg, May 2021, pp. 173–202. DOI: 10.1007/978-3-030-75248-4_7.

[Bou00] Fabrice Boudot. "Efficient Proofs that a Committed Number Lies in an Interval". In: *Advances in Cryptology – EUROCRYPT 2000.* Ed. by Bart Preneel. Vol. 1807. Lecture Notes in Computer Science. Springer, Heidelberg, May 2000, pp. 431–444. DOI: 10.1007/3-540-45539-6_31.

[BR93] Mihir Bellare and Phillip Rogaway. "Random Oracles are Practical: A Paradigm for Designing Efficient Protocols". In: *ACM CCS 93: 1st Conference on Computer and Communications Security.* Ed. by Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby. ACM Press, Nov. 1993, pp. 62–73. DOI: 10.1145/168588.168596.

[BR96] Mihir Bellare and Phillip Rogaway. "The Exact Security of Digital Signatures: How to Sign with RSA and Rabin". In: *Advances in Cryptology – EUROCRYPT'96.* Ed. by Ueli M. Maurer. Vol. 1070. Lecture Notes in Computer Science. Springer, Heidelberg, May 1996, pp. 399–416. DOI: 10.1007/3-540-68339-9_34.

[Bra00] Stefan Brands. *Rethinking public key infrastructures and digital certificates: building in privacy.* MIT Press, 2000.

[BS20] Dan Boneh and Victor Shoup. *A Graduate Course in Applied Cryptography.* Version 0.5. 2020. URL: https://toc.cryptobook.us/.

[BSCR+18] Eli Ben-Sasson, Alessandro Chiesa, Michael Riabzev, Nicholas Spooner, Madars Virza, and Nicholas P. Ward. *Aurora: Transparent Succinct Arguments for R1CS.* Cryptology ePrint Archive, Report 2018/828. https://eprint.iacr.org/2018/828. 2018.

[BT06] Andrej Bogdanov and Luca Trevisan. "Average-Case Complexity". In: *Foundations and Trends in Theoretical Computer Science* 2.1 (2006). DOI: 10.1561/0400000004. URL: https://doi.org/10.1561/0400000004.

[Can00] Ran Canetti. “Security and Composition of Multiparty Cryptographic Protocols”. In: *Journal of Cryptology* 13.1 (Jan. 2000), pp. 143–202. DOI: 10.1007/s001459910006.

[CCH+19] Ran Canetti, Yilei Chen, Justin Holmgren, Alex Lombardi, Guy N. Rothblum, Ron D. Rothblum, and Daniel Wichs. “Fiat-Shamir: from practice to theory”. In: *51st Annual ACM Symposium on Theory of Computing*. Ed. by Moses Charikar and Edith Cohen. ACM Press, June 2019, pp. 1082–1090. DOI: 10.1145/3313276.3316380.

[CCs08] Jan Camenisch, Rafik Chaabouni, and abhi shelat. “Efficient Protocols for Set Membership and Range Proofs”. In: *Advances in Cryptology – ASIACRYPT 2008*. Ed. by Josef Pieprzyk. Vol. 5350. Lecture Notes in Computer Science. Springer, Heidelberg, Dec. 2008, pp. 234–252. DOI: 10.1007/978-3-540-89255-7_15.

[CD98] Ronald Cramer and Ivan Damgård. “Zero-Knowledge Proofs for Finite Field Arithmetic; or: Can Zero-Knowledge Be for Free?” In: *Advances in Cryptology – CRYPTO’98*. Ed. by Hugo Krawczyk. Vol. 1462. Lecture Notes in Computer Science. Springer, Heidelberg, Aug. 1998, pp. 424–441. DOI: 10.1007/BFb0055745.

[CDG+17] Melissa Chase, David Derler, Steven Goldfeder, Claudio Orlandi, Sebastian Ramacher, Christian Rechberger, Daniel Slamanig, and Greg Zaverucha. “Post-Quantum Zero-Knowledge and Signatures from Symmetric-Key Primitives”. In: *ACM CCS 2017: 24th Conference on Computer and Communications Security*. Ed. by Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu. ACM Press, 2017, pp. 1825–1842. DOI: 10.1145/3133956.3133997.

[CDVV14] Siu on Chan, Ilias Diakonikolas, Paul Valiant, and Gregory Valiant. “Optimal Algorithms for Testing Closeness of Discrete Distributions”. In: *25th Annual ACM-SIAM Symposium on Discrete Algorithms*. Ed. by Chandra Chekuri. ACM-SIAM, Jan. 2014, pp. 1193–1203. DOI: 10.1137/1.9781611973402.88.

[CFF+21] Matteo Campanelli, Antonio Faonio, Dario Fiore, Anaïs Querol, and Hadrián Rodríguez. “Lunar: A Toolbox for More Efficient Universal and Updatable zkSNARKs and Commit-and-Prove Extensions”. In: *Advances in Cryptology – ASIACRYPT 2021, Part III*. Ed. by Mehdi Tibouchi and Huaxiong Wang. Vol. 13092. Lecture Notes in Computer Science. Springer, Heidelberg, Dec. 2021, pp. 3–33. DOI: 10.1007/978-3-030-92078-4_1.

[CFT98] Agnes Hui Chan, Yair Frankel, and Yiannis Tsiounis. “Easy Come - Easy Go Divisible Cash”. In: *Advances in Cryptology – EUROCRYPT’98*. Ed. by Kaisa Nyberg. Vol. 1403. Lecture Notes in Computer Science. Springer, Heidelberg, 1998, pp. 561–575. DOI: 10.1007/BFb0054154.

[CGH98] Ran Canetti, Oded Goldreich, and Shai Halevi. “The Random Oracle Methodology, Revisited (Preliminary Version)”. In: *30th Annual ACM Symposium on Theory of Computing*. ACM Press, May 1998, pp. 209–218. DOI: 10.1145/276698.276741.

[CGKR22a] Geoffroy Couteau, Dahmun Goudarzi, Michael Klooß, and Michael Reichle. “Sharp: Short Relaxed Range Proofs”. In: *CCS ’22: 2022 ACM SIGSAC Conference on Computer and Communications Security, Los Angeles, CA, USA, November 7 - 11, 2022*. Ed. by Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi. To appear. ACM, 2022. DOI: 10.1145/3548606.3560628. URL: https://doi.org/10.1145/3548606.3560628.

[CGKR22b] Geoffroy Couteau, Dahmun Goudarzi, Michael Klooß, and Michael Reichle. *Sharp: Short Relaxed Range Proofs*. Cryptology ePrint Archive, Report 2022/1153. https://eprint.iacr.org/2022/1153. 2022.

[Cha82] David Chaum. “Blind Signatures for Untraceable Payments”. In: *Advances in Cryptology – CRYPTO’82*. Ed. by David Chaum, Ronald L. Rivest, and Alan T. Sherman. Plenum Press, New York, USA, 1982, pp. 199–203.

[Cha90] David Chaum. “Showing Credentials without Identification Transferring Signatures between Unconditionally Unlinkable Pseudonyms”. In: *Advances in Cryptology – AUSCRYPT’90*. Ed. by Jennifer Seberry and Josef Pieprzyk. Vol. 453. Lecture Notes in Computer Science. Springer, Heidelberg, Jan. 1990, pp. 246–264. DOI: `10.1007/BFb0030366`.

[CHJ+22] HeeWon Chung, Kyoohyung Han, Chanyang Ju, Myungsun Kim, and Jae Hong Seo. “Bulletproofs+: Shorter Proofs for a Privacy-Enhanced Distributed Ledger”. In: *IEEE Access* 10 (2022), pp. 42067–42082. DOI: `10.1109/ACCESS.2022.3167806`. URL: `https://doi.org/10.1109/ACCESS.2022.3167806`.

[CHL05] Jan Camenisch, Susan Hohenberger, and Anna Lysyanskaya. “Compact E-Cash”. In: *Advances in Cryptology – EUROCRYPT 2005*. Ed. by Ronald Cramer. Vol. 3494. Lecture Notes in Computer Science. Springer, Heidelberg, May 2005, pp. 302–321. DOI: `10.1007/11426639_18`.

[CHM+20] Alessandro Chiesa, Yuncong Hu, Mary Maller, Pratyush Mishra, Noah Vesely, and Nicholas P. Ward. “Marlin: Preprocessing zkSNARKs with Universal and Updatable SRS”. In: *Advances in Cryptology – EUROCRYPT 2020, Part I*. Ed. by Anne Canteaut and Yuval Ishai. Vol. 12105. Lecture Notes in Computer Science. Springer, Heidelberg, May 2020, pp. 738–768. DOI: `10.1007/978-3-030-45721-1_26`.

[CKLM12] Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Sarah Meiklejohn. “Malleable Proof Systems and Applications”. In: *Advances in Cryptology – EUROCRYPT 2012*. Ed. by David Pointcheval and Thomas Johansson. Vol. 7237. Lecture Notes in Computer Science. Springer, Heidelberg, Apr. 2012, pp. 281–300. DOI: `10.1007/978-3-642-29011-4_18`.

[CKLR21a] Geoffroy Couteau, Michael Klooß, Huang Lin, and Michael Reichle. *Efficient Range Proofs with Transparent Setup from Bounded Integer Commitments*. Cryptology ePrint Archive, Report 2021/540. `https://eprint.iacr.org/2021/540`. 2021.

[CKLR21b] Geoffroy Couteau, Michael Klooß, Huang Lin, and Michael Reichle. “Efficient Range Proofs with Transparent Setup from Bounded Integer Commitments”. In: *Advances in Cryptology – EUROCRYPT 2021, Part III*. Ed. by Anne Canteaut and François-Xavier Standaert. Vol. 12698. Lecture Notes in Computer Science. Springer, Heidelberg, Oct. 2021, pp. 247–277. DOI: `10.1007/978-3-030-77883-5_9`.

[CKN03] Ran Canetti, Hugo Krawczyk, and Jesper Buus Nielsen. “Relaxing Chosen-Ciphertext Security”. In: *Advances in Cryptology – CRYPTO 2003*. Ed. by Dan Boneh. Vol. 2729. Lecture Notes in Computer Science. Springer, Heidelberg, Aug. 2003, pp. 565–582. DOI: `10.1007/978-3-540-45146-4_33`.

[CL01] Jan Camenisch and Anna Lysyanskaya. “An Efficient System for Non-transferable Anonymous Credentials with Optional Anonymity Revocation”. In: *Advances in Cryptology – EUROCRYPT 2001*. Ed. by Birgit Pfitzmann. Vol. 2045. Lecture Notes in Computer Science. Springer, Heidelberg, May 2001, pp. 93–118. DOI: `10.1007/3-540-44987-6_7`.

[CL03] Jan Camenisch and Anna Lysyanskaya. “A Signature Scheme with Efficient Protocols”. In: *SCN 02: 3rd International Conference on Security in Communication Networks*. Ed. by Stelvio Cimato, Clemente Galdi, and Giuseppe Persiano. Vol. 2576. Lecture Notes in Computer Science. Springer, Heidelberg, Sept. 2003, pp. 268–289. DOI: `10.1007/3-540-36413-7_20`.

[CL15] Guilhem Castagnos and Fabien Laguillaumie. *Linearly Homomorphic Encryption from DDH.* Cryptology ePrint Archive, Report 2015/047. `https://eprint.iacr.org/2015/047`. 2015.

[CLP15] Kai-Min Chung, Edward Lui, and Rafael Pass. “From Weak to Strong Zero-Knowledge and Applications”. In: *TCC 2015: 12th Theory of Cryptography Conference, Part I.* Ed. by Yevgeniy Dodis and Jesper Buus Nielsen. Vol. 9014. Lecture Notes in Computer Science. Springer, Heidelberg, Mar. 2015, pp. 66–92. DOI: `10.1007/978-3-662-46494-6_4`.

[CMS19] Alessandro Chiesa, Peter Manohar, and Nicholas Spooner. “Succinct Arguments in the Quantum Random Oracle Model”. In: *TCC 2019: 17th Theory of Cryptography Conference, Part II.* Ed. by Dennis Hofheinz and Alon Rosen. Vol. 11892. Lecture Notes in Computer Science. Springer, Heidelberg, Dec. 2019, pp. 1–29. DOI: `10.1007/978-3-030-36033-7_1`.

[CMZ14] Melissa Chase, Sarah Meiklejohn, and Greg Zaverucha. “Algebraic MACs and Keyed-Verification Anonymous Credentials”. In: *ACM CCS 2014: 21st Conference on Computer and Communications Security.* Ed. by Gail-Joon Ahn, Moti Yung, and Ninghui Li. ACM Press, Nov. 2014, pp. 1205–1216. DOI: `10.1145/2660267.2660328`.

[CPP17] Geoffroy Couteau, Thomas Peters, and David Pointcheval. “Removing the Strong RSA Assumption from Arguments over the Integers”. In: *Advances in Cryptology – EUROCRYPT 2017, Part II.* Ed. by Jean-Sébastien Coron and Jesper Buus Nielsen. Vol. 10211. Lecture Notes in Computer Science. Springer, Heidelberg, 2017, pp. 321–350. DOI: `10.1007/978-3-319-56614-6_11`.

[CPs07] Ran Canetti, Rafael Pass, and abhi shelat. “Cryptography from Sunspots: How to Use an Imperfect Reference String”. In: *48th Annual Symposium on Foundations of Computer Science.* IEEE Computer Society Press, Oct. 2007, pp. 249–259. DOI: `10.1109/FOCS.2007.70`.

[CR19] Geoffroy Couteau and Michael Reichle. “Non-interactive Keyed-Verification Anonymous Credentials”. In: *PKC 2019: 22nd International Conference on Theory and Practice of Public Key Cryptography, Part I.* Ed. by Dongdai Lin and Kazue Sako. Vol. 11442. Lecture Notes in Computer Science. Springer, Heidelberg, Apr. 2019, pp. 66–96. DOI: `10.1007/978-3-030-17253-4_3`.

[DF02] Ivan Damgård and Eiichiro Fujisaki. “A Statistically-Hiding Integer Commitment Scheme Based on Groups with Hidden Order”. In: *Advances in Cryptology – ASIACRYPT 2002.* Ed. by Yuliang Zheng. Vol. 2501. Lecture Notes in Computer Science. Springer, Heidelberg, Dec. 2002, pp. 125–142. DOI: `10.1007/3-540-36178-2_8`.

[DFGK14] George Danezis, Cédric Fournet, Jens Groth, and Markulf Kohlweiss. “Square Span Programs with Applications to Succinct NIZK Arguments”. In: *Advances in Cryptology – ASIACRYPT 2014, Part I.* Ed. by Palash Sarkar and Tetsu Iwata. Vol. 8873. Lecture Notes in Computer Science. Springer, Heidelberg, Dec. 2014, pp. 532–550. DOI: `10.1007/978-3-662-45611-8_28`.

[DG12] Ning Ding and Dawu Gu. “On Constant-Round Precise Zero-Knowledge”. In: *ICICS 12: 14th International Conference on Information and Communication Security.* Ed. by Tat Wing Chim and Tsz Hon Yuen. Vol. 7618. Lecture Notes in Computer Science. Springer, Heidelberg, Oct. 2012, pp. 178–190. DOI: `10.1007/978-3-642-34129-8_16`.

[DGS21] Samuel Dobson, Steven Galbraith, and Benjamin Smith. “Trustless unknown-order groups”. In: 2021. URL: `https://eprint.iacr.org/2020/196`.

[DNRS03] Cynthia Dwork, Moni Naor, Omer Reingold, and Larry J. Stockmeyer. "Magic Functions". In: *J. ACM* 50.6 (2003), pp. 852–921. DOI: `10.1145/950620.950623`. URL: `https://doi.org/10.1145/950620.950623`.

[EG14] Alex Escala and Jens Groth. "Fine-Tuning Groth-Sahai Proofs". In: *PKC 2014: 17th International Conference on Theory and Practice of Public Key Cryptography*. Ed. by Hugo Krawczyk. Vol. 8383. Lecture Notes in Computer Science. Springer, Heidelberg, Mar. 2014, pp. 630–649. DOI: `10.1007/978-3-642-54631-0_36`.

[EHK+13] Alex Escala, Gottfried Herold, Eike Kiltz, Carla Ràfols, and Jorge Villar. "An Algebraic Framework for Diffie-Hellman Assumptions". In: *Advances in Cryptology – CRYPTO 2013, Part II*. Ed. by Ran Canetti and Juan A. Garay. Vol. 8043. Lecture Notes in Computer Science. Springer, Heidelberg, Aug. 2013, pp. 129–147. DOI: `10.1007/978-3-642-40084-1_8`.

[EPRS17] Adam Everspaugh, Kenneth G. Paterson, Thomas Ristenpart, and Samuel Scott. "Key Rotation for Authenticated Encryption". In: *Advances in Cryptology – CRYPTO 2017, Part III*. Ed. by Jonathan Katz and Hovav Shacham. Vol. 10403. Lecture Notes in Computer Science. Springer, Heidelberg, Aug. 2017, pp. 98–129. DOI: `10.1007/978-3-319-63697-9_4`.

[FBK+21] Sebastian H. Faller, Pascal Baumer, Michael Klooß, Alexander Koch, Astrid Ottenhues, and Markus Raiber. "Black-Box Accumulation Based on Lattices". In: *18th IMA International Conference on Cryptography and Coding*. Ed. by Maura B. Paterson. Vol. 13129. Lecture Notes in Computer Science. Springer, Heidelberg, Dec. 2021, pp. 220–246. DOI: `10.1007/978-3-030-92641-0_11`.

[Fei90] Uriel Feige. "Alternative models for zero-knowledge interactive proofs". PhD thesis. Weizmann Institute of Science, 1990.

[FKL18] Georg Fuchsbauer, Eike Kiltz, and Julian Loss. "The Algebraic Group Model and its Applications". In: *Advances in Cryptology – CRYPTO 2018, Part II*. Ed. by Hovav Shacham and Alexandra Boldyreva. Vol. 10992. Lecture Notes in Computer Science. Springer, Heidelberg, Aug. 2018, pp. 33–62. DOI: `10.1007/978-3-319-96881-0_2`.

[FO97] Eiichiro Fujisaki and Tatsuaki Okamoto. "Statistical Zero Knowledge Protocols to Prove Modular Polynomial Relations". In: *Advances in Cryptology – CRYPTO'97*. Ed. by Burton S. Kaliski Jr. Vol. 1294. Lecture Notes in Computer Science. Springer, Heidelberg, Aug. 1997, pp. 16–30. DOI: `10.1007/BFb0052225`.

[FO99] Eiichiro Fujisaki and Tatsuaki Okamoto. "Secure Integration of Asymmetric and Symmetric Encryption Schemes". In: *Advances in Cryptology – CRYPTO'99*. Ed. by Michael J. Wiener. Vol. 1666. Lecture Notes in Computer Science. Springer, Heidelberg, Aug. 1999, pp. 537–554. DOI: `10.1007/3-540-48405-1_34`.

[FS87] Amos Fiat and Adi Shamir. "How to Prove Yourself: Practical Solutions to Identification and Signature Problems". In: *Advances in Cryptology – CRYPTO'86*. Ed. by Andrew M. Odlyzko. Vol. 263. Lecture Notes in Computer Science. Springer, Heidelberg, Aug. 1987, pp. 186–194. DOI: `10.1007/3-540-47721-7_12`.

[FSW03] Pierre-Alain Fouque, Jacques Stern, and Jan-Geert Wackers. "CryptoComputing with Rationals". In: *FC 2002: 6th International Conference on Financial Cryptography*. Ed. by Matt Blaze. Vol. 2357. Lecture Notes in Computer Science. Springer, Heidelberg, Mar. 2003, pp. 136–146.

[Gen09] Craig Gentry. “Fully homomorphic encryption using ideal lattices”. In: *41st Annual ACM Symposium on Theory of Computing*. Ed. by Michael Mitzenmacher. ACM Press, 2009, pp. 169–178. DOI: 10.1145/1536414.1536440.

[GGPR13] Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. “Quadratic Span Programs and Succinct NIZKs without PCPs”. In: *Advances in Cryptology – EUROCRYPT 2013*. Ed. by Thomas Johansson and Phong Q. Nguyen. Vol. 7881. Lecture Notes in Computer Science. Springer, Heidelberg, May 2013, pp. 626–645. DOI: 10.1007/978-3-642-38348-9_37.

[GI08] Jens Groth and Yuval Ishai. “Sub-linear Zero-Knowledge Argument for Correctness of a Shuffle”. In: *Advances in Cryptology – EUROCRYPT 2008*. Ed. by Nigel P. Smart. Vol. 4965. Lecture Notes in Computer Science. Springer, Heidelberg, Apr. 2008, pp. 379–396. DOI: 10.1007/978-3-540-78967-3_22.

[GK03] Shafi Goldwasser and Yael Tauman Kalai. “On the (In)security of the Fiat-Shamir Paradigm”. In: *44th Annual Symposium on Foundations of Computer Science*. IEEE Computer Society Press, Oct. 2003, pp. 102–115. DOI: 10.1109/SFCS.2003.1238185.

[GK96] Oded Goldreich and Ariel Kahan. “How to Construct Constant-Round Zero-Knowledge Proof Systems for NP”. In: *Journal of Cryptology* 9.3 (June 1996), pp. 167–190.

[GKM+18] Jens Groth, Markulf Kohlweiss, Mary Maller, Sarah Meiklejohn, and Ian Miers. “Updatable and Universal Common Reference Strings with Applications to zk-SNARKs”. In: *Advances in Cryptology – CRYPTO 2018, Part III*. Ed. by Hovav Shacham and Alexandra Boldyreva. Vol. 10993. Lecture Notes in Computer Science. Springer, Heidelberg, Aug. 2018, pp. 698–728. DOI: 10.1007/978-3-319-96878-0_24.

[GM98] Oded Goldreich and Bernd Meyer. “Computational Indistinguishability: Algorithms vs. Circuits”. In: *Theor. Comput. Sci.* 191.1-2 (1998), pp. 215–218. DOI: 10.1016/S0304-3975(97)00162-X. URL: https://doi.org/10.1016/S0304-3975(97)00162-X.

[GMNO18] Rosario Gennaro, Michele Minelli, Anca Nitulescu, and Michele Orrù. “Lattice-Based zk-SNARKs from Square Span Programs”. In: *ACM CCS 2018: 25th Conference on Computer and Communications Security*. Ed. by David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang. ACM Press, Oct. 2018, pp. 556–573. DOI: 10.1145/3243734.3243845.

[GMO16] Irene Giacomelli, Jesper Madsen, and Claudio Orlandi. “ZKBoo: Faster Zero-Knowledge for Boolean Circuits”. In: *USENIX Security 2016: 25th USENIX Security Symposium*. Ed. by Thorsten Holz and Stefan Savage. USENIX Association, Aug. 2016, pp. 1069–1083.

[GMR85] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. “The Knowledge Complexity of Interactive Proof-Systems (Extended Abstract)”. In: *17th Annual ACM Symposium on Theory of Computing*. ACM Press, May 1985, pp. 291–304. DOI: 10.1145/22145.22178.

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. “The Knowledge Complexity of Interactive Proof Systems”. In: *SIAM Journal on Computing* 18.1 (1989), pp. 186–208.

[GMW86] Oded Goldreich, Silvio Micali, and Avi Wigderson. “Proofs that Yield Nothing But their Validity and a Methodology of Cryptographic Protocol Design (Extended Abstract)”. In: *27th Annual Symposium on Foundations of Computer Science*. IEEE Computer Society Press, Oct. 1986, pp. 174–187. DOI: 10.1109/SFCS.1986.47.

[Gol01] Oded Goldreich. *The Foundations of Cryptography - Volume 1: Basic Techniques*. Cambridge University Press, 2001. ISBN: 0-521-79172-3. DOI: 10.1017/CBO9780511546891. URL: http://www.wisdom.weizmann.ac.il/\%7Eoded/foc-vol1.html.

[Gol07] Oded Goldreich. "On Expected Probabilistic Polynomial-Time Adversaries: A Suggestion for Restricted Definitions and Their Benefits". In: *TCC 2007: 4th Theory of Cryptography Conference*. Ed. by Salil P. Vadhan. Vol. 4392. Lecture Notes in Computer Science. Springer, Heidelberg, Feb. 2007, pp. 174–193. DOI: 10.1007/978-3-540-70936-7_10.

[Gol10] Oded Goldreich. "On Expected Probabilistic Polynomial-Time Adversaries: A Suggestion for Restricted Definitions and Their Benefits". In: *Journal of Cryptology* 23.1 (Jan. 2010), pp. 1–36. DOI: 10.1007/s00145-009-9050-5.

[Gol11a] Oded Goldreich. "Average Case Complexity, Revisited". In: *Studies in Complexity and Cryptography*. Vol. 6650. Lecture Notes in Computer Science. Springer, 2011, pp. 422–450.

[Gol11b] Oded Goldreich. "Notes on Levin's Theory of Average-Case Complexity". In: *Studies in Complexity and Cryptography*. Vol. 6650. Lecture Notes in Computer Science. Springer, 2011, pp. 233–247.

[Gol93] Oded Goldreich. "A Uniform-Complexity Treatment of Encryption and Zero-Knowledge". In: *Journal of Cryptology* 6.1 (Mar. 1993), pp. 21–53. DOI: 10.1007/BF02620230.

[GOP+22] Chaya Ganesh, Claudio Orlandi, Mahak Pancholi, Akira Takahashi, and Daniel Tschudi. "Fiat-Shamir Bulletproofs are Non-Malleable (in the Algebraic Group Model)". In: *Advances in Cryptology – EUROCRYPT 2022, Part II*. Ed. by Orr Dunkelman and Stefan Dziembowski. Vol. 13276. Lecture Notes in Computer Science. Springer, Heidelberg, 2022, pp. 397–426. DOI: 10.1007/978-3-031-07085-3_14.

[GP22] Yao Jiang Galteland and Jiaxin Pan. *Backward-Leak Uni-Directional Updatable Encryption from Public Key Encryption*. Cryptology ePrint Archive, Report 2022/324. https://eprint.iacr.org/2022/324. 2022.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. "Trapdoors for hard lattices and new cryptographic constructions". In: *40th Annual ACM Symposium on Theory of Computing*. Ed. by Richard E. Ladner and Cynthia Dwork. ACM Press, May 2008, pp. 197–206. DOI: 10.1145/1374376.1374407.

[Gro05] Jens Groth. "Non-interactive Zero-Knowledge Arguments for Voting". In: *ACNS 05: 3rd International Conference on Applied Cryptography and Network Security*. Ed. by John Ioannidis, Angelos Keromytis, and Moti Yung. Vol. 3531. Lecture Notes in Computer Science. Springer, Heidelberg, June 2005, pp. 467–482. DOI: 10.1007/11496137_32.

[Gro09] Jens Groth. "Linear Algebra with Sub-linear Zero-Knowledge Arguments". In: *Advances in Cryptology – CRYPTO 2009*. Ed. by Shai Halevi. Vol. 5677. Lecture Notes in Computer Science. Springer, Heidelberg, Aug. 2009, pp. 192–208. DOI: 10.1007/978-3-642-03356-8_12.

[Gro11] Jens Groth. "Efficient Zero-Knowledge Arguments from Two-Tiered Homomorphic Commitments". In: *Advances in Cryptology – ASIACRYPT 2011*. Ed. by Dong Hoon Lee and Xiaoyun Wang. Vol. 7073. Lecture Notes in Computer Science. Springer, Heidelberg, Dec. 2011, pp. 431–448. DOI: 10.1007/978-3-642-25385-0_23.

[Gro16] Jens Groth. "On the Size of Pairing-Based Non-interactive Arguments". In: *Advances in Cryptology – EUROCRYPT 2016, Part II*. Ed. by Marc Fischlin and Jean-Sébastien Coron. Vol. 9666. Lecture Notes in Computer Science. Springer, Heidelberg, May 2016, pp. 305–326. DOI: 10.1007/978-3-662-49896-5_11.

[GS08] Jens Groth and Amit Sahai. “Efficient Non-interactive Proof Systems for Bilinear Groups”. In: *Advances in Cryptology – EUROCRYPT 2008*. Ed. by Nigel P. Smart. Vol. 4965. Lecture Notes in Computer Science. Springer, Heidelberg, Apr. 2008, pp. 415–432. DOI: 10.1007/978-3-540-78967-3_24.

[GS98] Oded Goldreich and Madhu Sudan. “Computational Indistinguishability: A Sample Hierarchy”. In: *Proceedings of the 13th Annual IEEE Conference on Computational Complexity, Buffalo, New York, USA, June 15-18, 1998*. IEEE Computer Society, 1998, pp. 24–33. DOI: 10.1109/CCC.1998.694588. URL: https://doi.org/10.1109/CCC.1998.694588.

[GT21] Ashrujit Ghoshal and Stefano Tessaro. “Tight State-Restoration Soundness in the Algebraic Group Model”. In: *Advances in Cryptology – CRYPTO 2021, Part III*. Ed. by Tal Malkin and Chris Peikert. Vol. 12827. Lecture Notes in Computer Science. Virtual Event: Springer, Heidelberg, Aug. 2021, pp. 64–93. DOI: 10.1007/978-3-030-84252-9_3.

[GW11] Craig Gentry and Daniel Wichs. “Separating succinct non-interactive arguments from all falsifiable assumptions”. In: *43rd Annual ACM Symposium on Theory of Computing*. Ed. by Lance Fortnow and Salil P. Vadhan. ACM Press, June 2011, pp. 99–108. DOI: 10.1145/1993636.1993651.

[GWC19] Ariel Gabizon, Zachary J. Williamson, and Oana Ciobotaru. *PLONK: Permutations over Lagrange-bases for Oecumenical Noninteractive arguments of Knowledge*. Cryptology ePrint Archive, Report 2019/953. https://eprint.iacr.org/2019/953. 2019.

[HG13] Ryan Henry and Ian Goldberg. “Batch Proofs of Partial Knowledge”. In: *ACNS 13: 11th International Conference on Applied Cryptography and Network Security*. Ed. by Michael J. Jacobson Jr., Michael E. Locasto, Payman Mohassel, and Reihaneh Safavi-Naini. Vol. 7954. Lecture Notes in Computer Science. Springer, Heidelberg, June 2013, pp. 502–517. DOI: 10.1007/978-3-642-38980-1_32.

[HHK+17] Gottfried Herold, Max Hoffmann, Michael Klooß, Carla Ràfols, and Andy Rupp. “New Techniques for Structural Batch Verification in Bilinear Groups with Applications to Groth-Sahai Proofs”. In: *ACM CCS 2017: 24th Conference on Computer and Communications Security*. Ed. by Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu. ACM Press, 2017, pp. 1547–1564. DOI: 10.1145/3133956.3134068.

[HHNR17] Gunnar Hartung, Max Hoffmann, Matthias Nagel, and Andy Rupp. “BBA+: Improving the Security and Applicability of Privacy-Preserving Point Collection”. In: *ACM CCS 2017: 24th Conference on Computer and Communications Security*. Ed. by Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu. ACM Press, 2017, pp. 1925–1942. DOI: 10.1145/3133956.3134071.

[HKR19a] Max Hoffmann, Michael Klooß, and Andy Rupp. “Efficient Zero-Knowledge Arguments in the Discrete Log Setting, Revisited”. In: *ACM CCS 2019: 26th Conference on Computer and Communications Security*. Ed. by Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz. ACM Press, Nov. 2019, pp. 2093–2110. DOI: 10.1145/3319535.3354251.

[HKR19b] Max Hoffmann, Michael Klooß, and Andy Rupp. *Efficient zero-knowledge arguments in the discrete log setting, revisited*. Cryptology ePrint Archive, Report 2019/944. https://eprint.iacr.org/2019/944. 2019.

[HKRR20] Max Hoffmann, Michael Klooß, Markus Raiber, and Andy Rupp. “Black-Box Wallets: Fast Anonymous Two-Way Payments for Constrained Devices”. In: *Proc. Priv. Enhancing Technol.* 2020.1 (2020), pp. 165–194. DOI: 10.2478/popets-2020-0010. URL: https://doi.org/10.2478/popets-2020-0010.

[HM98] Shai Halevi and Silvio Micali. *More on Proofs of Knowledge*. Cryptology ePrint Archive, Report 1998/015. https://eprint.iacr.org/1998/015. 1998.

[HUM13] Dennis Hofheinz, Dominique Unruh, and Jörn Müller-Quade. "Polynomial Runtime and Composability". In: *Journal of Cryptology* 26.3 (July 2013), pp. 375–441. DOI: 10.1007/s00145-012-9127-4.

[IKOS07] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. "Zero-knowledge from secure multiparty computation". In: *39th Annual ACM Symposium on Theory of Computing*. Ed. by David S. Johnson and Uriel Feige. ACM Press, June 2007, pp. 21–30. DOI: 10.1145/1250790.1250794.

[Jia20] Yao Jiang. "The Direction of Updatable Encryption Does Not Matter Much". In: *Advances in Cryptology – ASIACRYPT 2020, Part III*. Ed. by Shiho Moriai and Huaxiong Wang. Vol. 12493. Lecture Notes in Computer Science. Springer, Heidelberg, Dec. 2020, pp. 529–558. DOI: 10.1007/978-3-030-64840-4_18.

[JLS21] Aayush Jain, Huijia Lin, and Amit Sahai. "Indistinguishability obfuscation from well-founded assumptions". In: *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*. Ed. by Samir Khuller and Virginia Vassilevska Williams. ACM, 2021, pp. 60–73. DOI: 10.1145/3406325.3451093. URL: https://doi.org/10.1145/3406325.3451093.

[JR16] Tibor Jager and Andy Rupp. "Black-Box Accumulation: Collecting Incentives in a Privacy-Preserving Way". In: *Proc. Priv. Enhancing Technol.* 2016.3 (2016), pp. 62–82. DOI: 10.1515/popets-2016-0016. URL: https://doi.org/10.1515/popets-2016-0016.

[JT20] Joseph Jaeger and Stefano Tessaro. "Expected-Time Cryptography: Generic Techniques and Applications to Concrete Soundness". In: *TCC 2020: 18th Theory of Cryptography Conference, Part III*. Ed. by Rafael Pass and Krzysztof Pietrzak. Vol. 12552. Lecture Notes in Computer Science. Springer, Heidelberg, Nov. 2020, pp. 414–443. DOI: 10.1007/978-3-030-64381-2_15.

[KHRS21] Christiane Kuhn, Dennis Hofheinz, Andy Rupp, and Thorsten Strufe. "Onion Routing with Replies". In: *Advances in Cryptology – ASIACRYPT 2021, Part II*. Ed. by Mehdi Tibouchi and Huaxiong Wang. Vol. 13091. Lecture Notes in Computer Science. Springer, Heidelberg, Dec. 2021, pp. 573–604. DOI: 10.1007/978-3-030-92075-3_20.

[KK04] Takeshi Koshiba and Kaoru Kurosawa. "Short Exponent Diffie-Hellman Problems". In: *PKC 2004: 7th International Workshop on Theory and Practice in Public Key Cryptography*. Ed. by Feng Bao, Robert Deng, and Jianying Zhou. Vol. 2947. Lecture Notes in Computer Science. Springer, Heidelberg, Mar. 2004, pp. 173–186. DOI: 10.1007/978-3-540-24632-9_13.

[KKS22] Aggelos Kiayias, Markulf Kohlweiss, and Amirreza Sarencheh. "PEReDi: Privacy-Enhanced, Regulated and Distributed Central Bank Digital Currencies". In: *CCS '22: 2022 ACM SIGSAC Conference on Computer and Communications Security, Los Angeles, CA, USA, November 7 - 11, 2022*. Ed. by Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi. To appear. ACM, 2022.

[KL05] Jonathan Katz and Yehuda Lindell. "Handling Expected Polynomial-Time Strategies in Simulation-Based Security Proofs". In: *TCC 2005: 2nd Theory of Cryptography Conference*. Ed. by Joe Kilian. Vol. 3378. Lecture Notes in Computer Science. Springer, Heidelberg, Feb. 2005, pp. 128–149. DOI: 10.1007/978-3-540-30576-7_8.

[KL08] Jonathan Katz and Yehuda Lindell. “Handling Expected Polynomial-Time Strategies in Simulation-Based Security Proofs”. In: *Journal of Cryptology* 21.3 (July 2008), pp. 303–349. DOI: 10.1007/s00145-007-9004-8.

[Klo20] Michael Klooß. *On (expected polynomial) runtime in cryptography*. Cryptology ePrint Archive, Report 2020/809. https://eprint.iacr.org/2020/809. 2020.

[Klo21] Michael Klooß. “On Expected Polynomial Runtime in Cryptography”. In: *TCC 2021: 19th Theory of Cryptography Conference, Part I*. Ed. by Kobbi Nissim and Brent Waters. Vol. 13042. Lecture Notes in Computer Science. Springer, Heidelberg, Nov. 2021, pp. 558–590. DOI: 10.1007/978-3-030-90459-3_19.

[KLR19] Michael Klooß, Anja Lehmann, and Andy Rupp. “(R)CCA Secure Updatable Encryption with Integrity Protection”. In: *Advances in Cryptology – EUROCRYPT 2019, Part I*. Ed. by Yuval Ishai and Vincent Rijmen. Vol. 11476. Lecture Notes in Computer Science. Springer, Heidelberg, May 2019, pp. 68–99. DOI: 10.1007/978-3-030-17653-2_3.

[KM13] Neal Koblitz and Alfred Menezes. “Another look at non-uniformity”. In: *Groups Complexity Cryptology* 5.2 (2013), pp. 117–139. DOI: 10.1515/gcc-2013-0008. URL: https://doi.org/10.1515/gcc-2013-0008.

[KM15] Neal Koblitz and Alfred J. Menezes. “The random oracle model: a twenty-year retrospective”. In: *Des. Codes Cryptogr.* 77.2-3 (2015), pp. 587–610. DOI: 10.1007/s10623-015-0094-2. URL: https://doi.org/10.1007/s10623-015-0094-2.

[KMSV21] Markulf Kohlweiss, Mary Maller, Janno Siim, and Mikhail Volkhov. “Snarky Ceremonies”. In: *Advances in Cryptology – ASIACRYPT 2021, Part III*. Ed. by Mehdi Tibouchi and Huaxiong Wang. Vol. 13092. Lecture Notes in Computer Science. Springer, Heidelberg, Dec. 2021, pp. 98–127. DOI: 10.1007/978-3-030-92078-4_4.

[KP01] Joe Kilian and Erez Petrank. “Concurrent and resettable zero-knowledge in polyloalgorithm rounds”. In: *33rd Annual ACM Symposium on Theory of Computing*. ACM Press, July 2001, pp. 560–569. DOI: 10.1145/380752.380851.

[KZM+15] Ahmed Kosba, Zhichao Zhao, Andrew Miller, Yi Qian, Hubert Chan, Charalampos Papamanthou, Rafael Pass, abhi shelat, and Elaine Shi. *C∅C∅: A Framework for Building Composable Zero-Knowledge Proofs*. Cryptology ePrint Archive, Report 2015/1093. https://eprint.iacr.org/2015/1093. 2015.

[Lee21] Jonathan Lee. “Dory: Efficient, Transparent Arguments for Generalised Inner Products and Polynomial Commitments”. In: *TCC 2021: 19th Theory of Cryptography Conference, Part II*. Ed. by Kobbi Nissim and Brent Waters. Vol. 13043. Lecture Notes in Computer Science. Springer, Heidelberg, Nov. 2021, pp. 1–34. DOI: 10.1007/978-3-030-90453-1_1.

[Lev86] Leonid A. Levin. “Average Case Complete Problems”. In: *SIAM J. Comput.* 15.1 (1986), pp. 285–286. DOI: 10.1137/0215020. URL: https://doi.org/10.1137/0215020.

[Lin03] Yehuda Lindell. “Parallel Coin-Tossing and Constant-Round Secure Two-Party Computation”. In: *Journal of Cryptology* 16.3 (June 2003), pp. 143–184. DOI: 10.1007/s00145-002-0143-7.

[Lin13] Yehuda Lindell. “A Note on Constant-Round Zero-Knowledge Proofs of Knowledge”. In: *Journal of Cryptology* 26.4 (Oct. 2013), pp. 638–654. DOI: 10.1007/s00145-012-9132-7.

[Lin17] Yehuda Lindell. “How to Simulate It - A Tutorial on the Simulation Proof Technique”. In: *Tutorials on the Foundations of Cryptography*. Springer International Publishing, 2017, pp. 277–346.

[Lip03] Helger Lipmaa. “On Diophantine Complexity and Statistical Zero-Knowledge Arguments”. In: *Advances in Cryptology – ASIACRYPT 2003*. Ed. by Chi-Sung Laih. Vol. 2894. Lecture Notes in Computer Science. Springer, Heidelberg, 2003, pp. 398–415. DOI: 10.1007/978-3-540-40061-5_26.

[LM20] David Lanzenberger and Ueli Maurer. “Coupling of Random Systems”. In: *TCC 2020: 18th Theory of Cryptography Conference, Part III*. Ed. by Rafael Pass and Krzysztof Pietrzak. Vol. 12552. Lecture Notes in Computer Science. Springer, Heidelberg, Nov. 2020, pp. 207–240. DOI: 10.1007/978-3-030-64381-2_8.

[LMR19] Russell W. F. Lai, Giulio Malavolta, and Viktoria Ronge. “Succinct Arguments for Bilinear Group Arithmetic: Practical Structure-Preserving Cryptography”. In: *ACM CCS 2019: 26th Conference on Computer and Communications Security*. Ed. by Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz. ACM Press, Nov. 2019, pp. 2057–2074. DOI: 10.1145/3319535.3354262.

[LNP22] Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Maxime Plancon. “Lattice-Based Zero-Knowledge Proofs and Applications: Shorter, Simpler, and More General”. In: *Advances in Cryptology – CRYPTO 2022*. Ed. by Yevgeniy Dodis and Thomas Shrimpton. Springer Nature Switzerland, 2022.

[LNS20] Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Gregor Seiler. “Practical Lattice-Based Zero-Knowledge Proofs for Integer Relations”. In: *ACM CCS 2020: 27th Conference on Computer and Communications Security*. Ed. by Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna. ACM Press, Nov. 2020, pp. 1051–1070. DOI: 10.1145/3372297.3417894.

[LT18] Anja Lehmann and Björn Tackmann. “Updatable Encryption with Post-Compromise Security”. In: *Advances in Cryptology – EUROCRYPT 2018, Part III*. Ed. by Jesper Buus Nielsen and Vincent Rijmen. Vol. 10822. Lecture Notes in Computer Science. Springer, Heidelberg, 2018, pp. 685–716. DOI: 10.1007/978-3-319-78372-7_22.

[Lyu09] Vadim Lyubashevsky. “Fiat-Shamir with Aborts: Applications to Lattice and Factoring-Based Signatures”. In: *Advances in Cryptology – ASIACRYPT 2009*. Ed. by Mitsuru Matsui. Vol. 5912. Lecture Notes in Computer Science. Springer, Heidelberg, Dec. 2009, pp. 598–616. DOI: 10.1007/978-3-642-10366-7_35.

[Lyu12] Vadim Lyubashevsky. “Lattice Signatures without Trapdoors”. In: *Advances in Cryptology – EUROCRYPT 2012*. Ed. by David Pointcheval and Thomas Johansson. Vol. 7237. Lecture Notes in Computer Science. Springer, Heidelberg, Apr. 2012, pp. 738–755. DOI: 10.1007/978-3-642-29011-4_43.

[Mau05] Ueli M. Maurer. “Abstract Models of Computation in Cryptography (Invited Paper)”. In: *10th IMA International Conference on Cryptography and Coding*. Ed. by Nigel P. Smart. Vol. 3796. Lecture Notes in Computer Science. Springer, Heidelberg, Dec. 2005, pp. 1–12.

[Mau15] Ueli Maurer. “Zero-knowledge proofs of knowledge for group homomorphisms”. In: *Des. Codes Cryptogr.* 77.2-3 (2015), pp. 663–676. DOI: 10.1007/s10623-015-0103-5. URL: https://doi.org/10.1007/s10623-015-0103-5.

[MBKM19] Mary Maller, Sean Bowe, Markulf Kohlweiss, and Sarah Meiklejohn. “Sonic: Zero-Knowledge SNARKs from Linear-Size Universal and Updatable Structured Reference Strings”. In: *ACM CCS 2019: 26th Conference on Computer and Communications Security*. Ed. by Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz. ACM Press, Nov. 2019, pp. 2111–2128. DOI: 10.1145/3319535.3339817.

[Mey94] Bernd Meyer. “Constructive Separation of Classes of Indistinguishable Ensembles”. In: *Proceedings of the Ninth Annual Structure in Complexity Theory Conference, Amsterdam, The Netherlands, June 28 - July 1, 1994.* IEEE Computer Society, 1994, pp. 198–204. DOI: 10.1109/SCT.1994.315804. URL: https://doi.org/10.1109/SCT.1994.315804.

[Mon] The Monero Project. *Monero.* https://github.com/monero-project/monero. 2022.

[MOV96] Alfred Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography.* CRC Press, 1996. ISBN: 0-8493-8523-7. DOI: 10.1201/9781439821916. URL: http://cacr.uwaterloo.ca/hac/.

[MP06] Silvio Micali and Rafael Pass. “Local zero knowledge”. In: *38th Annual ACM Symposium on Theory of Computing.* Ed. by Jon M. Kleinberg. ACM Press, May 2006, pp. 306–315. DOI: 10.1145/1132516.1132561.

[MPW22] Peihan Miao, Sikhar Patranabis, and Gaven Watson. *Unidirectional Updatable Encryption and Proxy Re-encryption from DDH or LWE.* Cryptology ePrint Archive, Report 2022/311. https://eprint.iacr.org/2022/311. 2022.

[MRV16] Paz Morillo, Carla Ràfols, and Jorge Luis Villar. “The Kernel Matrix Diffie-Hellman Assumption”. In: *Advances in Cryptology – ASIACRYPT 2016, Part I.* Ed. by Jung Hee Cheon and Tsuyoshi Takagi. Vol. 10031. Lecture Notes in Computer Science. Springer, Heidelberg, Dec. 2016, pp. 729–758. DOI: 10.1007/978-3-662-53887-6_27.

[Nao03] Moni Naor. “On Cryptographic Assumptions and Challenges (Invited Talk)”. In: *Advances in Cryptology – CRYPTO 2003.* Ed. by Dan Boneh. Vol. 2729. Lecture Notes in Computer Science. Springer, Heidelberg, Aug. 2003, pp. 96–109. DOI: 10.1007/978-3-540-45146-4_6.

[Nef01] C. Andrew Neff. “A Verifiable Secret Shuffle and Its Application to e-Voting”. In: *ACM CCS 2001: 8th Conference on Computer and Communications Security.* Ed. by Michael K. Reiter and Pierangela Samarati. ACM Press, Nov. 2001, pp. 116–125. DOI: 10.1145/501983.502000.

[Nie02] Jesper Buus Nielsen. “Separating Random Oracle Proofs from Complexity Theoretic Proofs: The Non-committing Encryption Case”. In: *Advances in Cryptology – CRYPTO 2002.* Ed. by Moti Yung. Vol. 2442. Lecture Notes in Computer Science. Springer, Heidelberg, Aug. 2002, pp. 111–126. DOI: 10.1007/3-540-45708-9_8.

[Nis21] Ryo Nishimaki. *The Direction of Updatable Encryption Does Matter.* Cryptology ePrint Archive, Report 2021/221. https://eprint.iacr.org/2021/221. 2021.

[Pas06] Rafael Pass. “A precise computational approach to knowledge”. PhD thesis. Massachusetts Institute of Technology, Cambridge, MA, USA, 2006.

[PBD07] Kun Peng, Colin Boyd, and Ed Dawson. “Batch zero-knowledge proof and verification and its applications”. In: *ACM Trans. Inf. Syst. Secur.* 10.2 (2007), p. 6. DOI: 10.1145/1237500.1237502. URL: https://doi.org/10.1145/1237500.1237502.

[Ped92] Torben P. Pedersen. “Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing”. In: *Advances in Cryptology – CRYPTO’91.* Ed. by Joan Feigenbaum. Vol. 576. Lecture Notes in Computer Science. Springer, Heidelberg, Aug. 1992, pp. 129–140. DOI: 10.1007/3-540-46766-1_9.

[PGHR13] Bryan Parno, Craig Gentry, Jon Howell, and Mariana Raykova. *Pinocchio: Nearly Practical Verifiable Computation.* Cryptology ePrint Archive, Report 2013/279. https://eprint.iacr.org/2013/279. 2013.

[PS19] Paul Pollack and Peter Schorn. “Dirichlet’s proof of the three-square theorem: An algorithmic perspective”. In: *Math. Comput.* 88.316 (2019), pp. 1007–1019. DOI: 10.1090/mcom/3349. URL: https://doi.org/10.1090/mcom/3349.

[PTV14] Rafael Pass, Wei-Lung Dustin Tseng, and Muthuramakrishnan Venkitasubramaniam. “Concurrent Zero Knowledge, Revisited”. In: *Journal of Cryptology* 27.1 (Jan. 2014), pp. 45–66. DOI: 10.1007/s00145-012-9137-2.

[RAD78] Ronald L. Rivest, Len Adleman, and Michael L. Dertouzos. “On data banks and privacy homomorphisms”. In: *Foundations of secure computation* 4.11 (1978), pp. 169–180.

[Ros04] Alon Rosen. “A Note on Constant-Round Zero-Knowledge Proofs for NP”. In: *TCC 2004: 1st Theory of Cryptography Conference.* Ed. by Moni Naor. Vol. 2951. Lecture Notes in Computer Science. Springer, Heidelberg, Feb. 2004, pp. 191–202. DOI: 10.1007/978-3-540-24638-1_11.

[RRR16] Omer Reingold, Guy N. Rothblum, and Ron D. Rothblum. “Constant-round interactive proofs for delegating computation”. In: *48th Annual ACM Symposium on Theory of Computing.* Ed. by Daniel Wichs and Yishay Mansour. ACM Press, June 2016, pp. 49–62. DOI: 10.1145/2897518.2897652.

[RS86] Michael O. Rabin and Jeffery O. Shallit. “Randomized Algorithms in Number Theory”. In: vol. 39. S1. 1986, S239–S256. DOI: https://doi.org/10.1002/cpa.3160390713.

[RVH17] Sietse Ringers, Eric R. Verheul, and Jaap-Henk Hoepman. “An Efficient Self-blindable Attribute-Based Credential Scheme”. In: *FC 2017: 21st International Conference on Financial Cryptography and Data Security.* Ed. by Aggelos Kiayias. Vol. 10322. Lecture Notes in Computer Science. Springer, Heidelberg, Apr. 2017, pp. 3–20.

[RZ21] Carla Ràfols and Arantxa Zapico. “An Algebraic Framework for Universal and Updatable SNARKs”. In: *Advances in Cryptology – CRYPTO 2021, Part I.* Ed. by Tal Malkin and Chris Peikert. Vol. 12825. Lecture Notes in Computer Science. Virtual Event: Springer, Heidelberg, Aug. 2021, pp. 774–804. DOI: 10.1007/978-3-030-84242-0_27.

[Set20] Srinath Setty. “Spartan: Efficient and General-Purpose zkSNARKs Without Trusted Setup”. In: *Advances in Cryptology – CRYPTO 2020, Part III.* Ed. by Daniele Micciancio and Thomas Ristenpart. Vol. 12172. Lecture Notes in Computer Science. Springer, Heidelberg, Aug. 2020, pp. 704–737. DOI: 10.1007/978-3-030-56877-1_25.

[Sho97] Victor Shoup. “Lower Bounds for Discrete Logarithms and Related Problems”. In: *Advances in Cryptology – EUROCRYPT’97.* Ed. by Walter Fumy. Vol. 1233. Lecture Notes in Computer Science. Springer, Heidelberg, May 1997, pp. 256–266. DOI: 10.1007/3-540-69053-0_18.

[SS21] Daniel Slamanig and Christoph Striecks. *Puncture ’Em All: Updatable Encryption with No-Directional Key Updates and Expiring Ciphertexts.* Cryptology ePrint Archive, Report 2021/268. https://eprint.iacr.org/2021/268. 2021.

[TCLM21] Sri Aravinda Krishnan Thyagarajan, Guilhem Castagnos, Fabien Laguillaumie, and Giulio Malavolta. “Efficient CCA Timed Commitments in Class Groups”. In: *ACM CCS 2021: 28th Conference on Computer and Communications Security.* Ed. by Giovanni Vigna and Elaine Shi. ACM Press, Nov. 2021, pp. 2663–2684. DOI: 10.1145/3460120.3484773.

[TW10] Björn Terelius and Douglas Wikström. “Proofs of Restricted Shuffles”. In: *AFRICACRYPT 10: 3rd International Conference on Cryptology in Africa.* Ed. by Daniel J. Bernstein and Tanja Lange. Vol. 6055. Lecture Notes in Computer Science. Springer, Heidelberg, May 2010, pp. 100–113.

[VGT+19] Henry de Valence, Jack Grigg, George Tankersley, Filippo Valsorda, and Isis Lovecruft. *The ristretto255 group.* Tech. rep. IETF CFRG Internet Draft, 2019.

[Web] *What is Jubjub?* `https://z.cash/technology/jubjub`.

[Wik18] Douglas Wikström. *Special Soundness Revisited.* Cryptology ePrint Archive, Report 2018/1157. `https://eprint.iacr.org/2018/1157`. 2018.

[Wik21] Douglas Wikström. *Special Soundness in the Random Oracle Model.* Cryptology ePrint Archive, Report 2021/1265. `https://eprint.iacr.org/2021/1265`. 2021.

[WKDC22] Karl Wüst, Kari Kostiainen, Noah Delius, and Srdjan Capkun. "Platypus: A Central Bank Digital Currency with Unlinkable Transactions and Privacy Preserving Regulation". In: *CCS '22: 2022 ACM SIGSAC Conference on Computer and Communications Security, Los Angeles, CA, USA, November 7 - 11, 2022.* Ed. by Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi. To appear. ACM, 2022.

[WTs+18] Riad S. Wahby, Ioanna Tzialla, abhi shelat, Justin Thaler, and Michael Walfish. "Doubly-Efficient zkSNARKs Without Trusted Setup". In: *2018 IEEE Symposium on Security and Privacy.* IEEE Computer Society Press, May 2018, pp. 926–943. DOI: `10.1109/SP.2018.00060`.

[Wui18] Pieter Wuille. *libsecp256k1.* `https://github.com/bitcoin/secp256k1`. 2018.

[Yao86] Andrew Chi-Chih Yao. "How to Generate and Exchange Secrets (Extended Abstract)". In: *27th Annual Symposium on Foundations of Computer Science.* IEEE Computer Society Press, Oct. 1986, pp. 162–167. DOI: `10.1109/SFCS.1986.25`.

[Zca] The Zcash developers. *Zcash.* `https://github.com/zcash/zcash`. 2022.

[ZkpComref] ZKProof. *ZKProof Community Reference. Version 0.3.* July 2022. Updated versions at https://docs.zkproof.org/reference.

Part III.

Appendix

A. Appendix for Chapter 3

A.1. Preliminaries Continued

A.1.1. Groups of Hidden Order

Following assumptions are relevant in groups of hidden order. Note that we still use additive notation, even if multiplicative notation is more common for RSA groups.

Remark A.1.1. In a cyclic group $\langle G \rangle$ of unknown order, a random group element be approximated via xG for $x \xleftarrow{\$} [0, \dots, LU_{\mathsf{up}} - 1]$ and xG has statistical distance at most $1/L$ from a random group element. Indeed, at most $1/4L$ (due to [CL15]).

The ORD assumption ensures, that it is hard to find (a multiple of) the order of non-trivial elements.

Definition A.1.2 (ORD). The **order (ORD) assumption** holds for a given group $\mathbb{G}$ if for any PPT adversary $\mathcal{A}$, there is a negligible function negl, such that

$$\Pr\left[\begin{array}{c}(W, \alpha) \leftarrow \mathcal{A}(\mathbb{G}); W \in \mathbb{G} \setminus \{0\}; \\ 0 \neq |\alpha| < 2^{\mathsf{poly}(\lambda)} \colon \alpha W = 0\end{array}\right] \leq \mathsf{negl}(\lambda)$$

This probability defines the advantage $\mathsf{Adv}^{\mathrm{ORD}}_{\mathcal{A}}(\lambda)$ of $\mathcal{A}$ against ORD.

We use the adaption of ORD to the class group setting of [BFS20] with corrections from [CKLR21b]. It is believed to hold in suitable class groups of imaginary quadratic orders and the subgroup of quadratic residues QR_n in RSA groups. For 128 bits of security, class groups require a discriminant of size 1827 bits and RSA groups a modulus size of 3072 bits [BJS10; TCLM21]. Further, the representation of class group elements can be compressed to 3/4 the size of the discriminant [DGS21].

Definition A.1.3 (e-fROOT). The **e-fractional root (e-fROOT) assumption** holds for group $\mathbb{G}$ if for any PPT adversary $\mathcal{A}$, there is a negligible function negl, such that

$$\Pr\left[\begin{array}{c}G \xleftarrow{\$} \mathbb{G}; (\alpha, \beta, U) \leftarrow \mathcal{A}(\mathbb{G}, G); U \in \mathbb{G}; \\ 0 \neq |\alpha| < 2^{\mathsf{poly}(\lambda)} \in \mathbb{Z}; |\beta| < 2^{\mathsf{poly}(\lambda)} \in \mathbb{Z} \colon \\ \beta U = \alpha G \wedge \frac{\beta}{\gcd(\alpha, \beta)} \neq e^k \text{ for } k \in \mathbb{N}\end{array}\right] \leq \mathsf{negl}(\lambda)$$

This probability defines the advantage $\mathsf{Adv}^{e\text{-fROOT}}_{\mathcal{A}}(\lambda)$ of $\mathcal{A}$ against e-fROOT.

The e-strong RSA assumption is defined as e-fROOT but where $\alpha = 1$ must hold. [BFS20] define this and show that e-strong RSA and ORD imply e-fROOT. The e-fROOT assumption clearly implies e-strong RSA and almost implies ORD, except for elements W with $e^k W = 0$.

The 1-fROOT assumption is equivalent to the (usual) strong RSA assumption and believed to hold in QR_n. The 2-fROOT assumption is believed to hold in suitable class groups of imaginary quadratic orders.

Remark A.1.4. Let $\mathbb{G}$ be a group, let $G \in \mathbb{G}$, and let (α, β, W) with $\alpha G = \beta W$. Then:

1. The e-fROOT experiment is won with (α, β, W) iff $\frac{\alpha}{\beta} \notin \mathbb{Z}[1/e]$.
2. The ORD experiment is won if $d = \gcd(\alpha, \beta)$ (or more generally any divisor d of α and β), we have $\frac{\alpha}{d} G \neq \frac{\beta}{d} W$.

The SI assumption ensures that random elements in the subgroup $\langle G \rangle$ are indistinguishable from random elements in $\mathbb{G}$.

Definition A.1.5 (SI). The **subgroup indistinguishability (SI) assumption** holds for group $\mathbb{G}$ if for any PPT adversary $\mathcal{A}$, there is a negligible function negl, such that

$$\Pr\left[\begin{array}{c} G, H_0 \xleftarrow{\$} \mathbb{G}, H_1 \xleftarrow{\$} \langle G \rangle \,; \\ b \xleftarrow{\$} \{0,1\}, b' \leftarrow \mathcal{A}(\mathbb{G}, G, H_b) : \\ b = b' \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

The left hand side defines the advantage $\mathsf{Adv}^{\mathrm{si}}_{\mathcal{A}}(\lambda)$ of $\mathcal{A}$ against SI.

Again, we use the adaption from [BFS20] of the SI assumption introduced in [BG10]. It is believed to hold in QR_n or suitable class groups of imaginary quadratic orders.

Definition A.1.6 ((D, e, N)-relaxed DLOG-relation). Let $\mathbb{G}$ be a group, $D, e, N \in \mathbb{N}$, and $\vec{G} = (G_0, \dots, G_N) \in \mathbb{G}^{N+1}$. Define the (D, e, N)**-relaxed DLOG relation w.r.t.** $\vec{G}$ as

$$\mathcal{R}_{D,e,N}(\vec{G}) = \left\{ (C, d, \{m_i\}_{i=1}^{N}) \;\middle|\; \begin{array}{c} dC = \sum_{i=0}^{N} m_i G_i \;\wedge\; \exists i \colon \frac{m_i}{d} \notin \mathbb{Z}[1/e] \\ \wedge\, d \in [0, D] \;\wedge\; m_i \in \mathbb{Z} \end{array} \right\}$$

The advantage $\mathsf{Adv}^{\text{rlx-dlog}}_{\mathbb{G},(D,e,N),\mathcal{A}}(\lambda)$ of $\mathcal{A}$ against the hardness of the (D, e, N)-relaxed DLOG-relation **with subgroup setup** (and without public coins), is defined as the following probability:

$$\Pr\left[\begin{array}{c} \mathbb{G} \leftarrow \mathsf{GrpGen}(1^\lambda); G_0 \xleftarrow{\$} \mathbb{G}; G_1, \dots, G_N \xleftarrow{\$} \langle G_0 \rangle \\ (C, d, m_0, \dots, m_N) \leftarrow \mathcal{A}(\mathbb{G}, G_0, \dots, G_N) : \\ (C, d, m_0, \dots, m_N) \in \mathcal{R}_{D,e,N}(\vec{G}) \end{array}\right].$$

We say that finding (D, e, N)-relaxed DLOG-relations **with subgroup setup** is **hard** in $\mathbb{G}$, if for every PPT adversary, there exists a negligible function negl such that $\mathsf{Adv}^{\text{rlx-dlog}}_{\mathbb{G},(D,e,N),\mathcal{A}}(\lambda) \leq \mathsf{negl}(\lambda)$.

We define hardness **with random setup** analogously, except that $G_i \xleftarrow{\$} \mathbb{G}$ for all i (instead of $G_i \xleftarrow{\$} \langle G_0 \rangle$).

We abbreviate $(D, e, 1)$-relaxed by (D, e)-relaxed.

Viewing C as a commitment and $(G_0, \dots, G_N)$ as a commitment key in Definition A.1.6 (which is exactly how we use it), (D, e, N)-relaxed DLog-relation hardness roughly holds if it is not possibly to open C to anything but an element in $\mathbb{Z}[1/e]$ (where we neglect the condition that $d \leq D$). The choice $N = 1$ is the most important one for (D, e, N)-relaxed DLOG-relations, as it is required for our applications and (as we will see) is equivalent $N > 1$. The $(D, e, 0)$-relaxed DLOG-relation is a (presumably) slightly weaker assumption, but is implied under additional restrictions (or assumptions).

Lemma A.1.7. *Let $\mathbb{G}$ be a group and let $\mathcal{A}$ be an algorithm. Then for* ***hardness with subgroup setup****, we have following implications.*

1. *The (D, e, N)-relaxed DLOG-relation tightly implies the (D, e, n)-relaxed DLOG-relation for any $n \leq N$.*
2. *The $(D, e, 1)$-relaxed DLOG-relation tightly implies the (D, e, N)-relaxed DLOG-relation for $N \in \mathbb{N}_0$.*
3. *The e-fROOT assumption tightly implies hardness of $(D, e, 0)$-relaxed DLOG-relation. If $D = \infty$, the assumptions are equivalent.*
4. *If the order $|\mathbb{G}|$ has no prime factors smaller than or equal to D, then $(D, e, 0)$-relaxed DLOG-relation tightly implies the $(D, e, 1)$-relaxed DLOG-relation.*

*Under the SI assumption in $\mathbb{G}$, the claims also hold for **hardness with random setup**.*

Item 4 is the general formulation to be used with $C(\lambda)$-rough groups [DF02], i.e. groups which have no subgroups of order smaller than $C(\lambda)$. The proof of item 4 uses the argument from [CKLR21b] which is an adaption of [DF02]. For item 2, a standard randomization technique is used (which is also used to show the tight equivalence of DLOG and DLOG-relations). Items 1 and 3 are immediate and included for completeness. The full proof of Lemma A.1.7 is relegated to Appendix A.1.3.

A.1.2. Transparent Setup and Assumptions Without Invertible Sampling

As we saw in Section 2.3.2.1, invertible sampling (a.k.a. reverse sampling) is known for many groups. Unfortunately, invertible sampling is not known to be possible in class groups (at the time of writing). This was first pointed out by [ADOS22] w.r.t. CKLR proofs [CKLR21b], but also affects our setting. CKLR resolved the problem by relying on ElGamal commitments and the DXDH assumption [ADOS22] which sacrifices efficiency. We rely on novel assumptions which take this into account by providing the adversary with sampling randomness. We stress that, while these assumptions are stronger than their counterparts, like DXDH they are all very plausible.

In the following, we denote by Sample the sampling algorithm and write $(G, \rho) \xleftarrow{\$} \mathsf{Sample}(1^\lambda, \mathbb{G})$, i.e. we consider the random coins ρ as an output, partially adopting the notation of [ADOS22]. This is more convenient and more flexible as it allows us to model leakage other than the random coins as well. We modify our assumptions to account for leakage of ρ to the adversary. The modifications are straightforward, but we provide them for completeness. Changes are highlighted in red.

Definition A.1.8 (S-Bounded DLSE and SEI w.r.t. Sample). Consider a group $\mathbb{G}$. The S-bounded **discrete logarithm with short exponents** (DLSE) assumption w.r.t. Sample holds if for all PPT $\mathcal{A}$ there is a negligible function negl such that

$$\Pr\left[\begin{matrix} (G, \rho) \leftarrow \mathsf{Sample}(1^\lambda, \mathbb{G}); \\ z \xleftarrow{\$} [0, S];\ z' \leftarrow \mathcal{A}(G, \rho, zG) \end{matrix} : z = z'\right] \leq \mathsf{negl}(\lambda)$$

The S-bounded **short exponent indistinguishability** (SEI) assumption w.r.t. Sample holds if for all PPT $\mathcal{A}$ there is a negligible negl function such that

$$\begin{aligned} &\Pr\left[(G, \rho) \leftarrow \mathsf{Sample}(1^\lambda, \mathbb{G}); z \xleftarrow{\$} [0, S] : \mathcal{A}(G, \rho, zG) = 1\right] \\ -&\Pr\left[(G, \rho) \leftarrow \mathsf{Sample}(1^\lambda, \mathbb{G}); z \xleftarrow{\$} \mathbb{Z}_{\mathrm{ord}(G)} : \mathcal{A}(G, \rho, zG) = 1\right] \\ \leq&\mathsf{negl}(\lambda) \end{aligned}$$

Definition A.1.9 (e-fROOT w.r.t. Sample). The **e-fractional root (e-fROOT) assumption w.r.t.** Sample holds for group $\mathbb{G}$ if for any PPT adversary $\mathcal{A}$, there is a negligible function negl, such that

$$\Pr\left[\begin{array}{c} (G,\rho) \leftarrow \mathsf{Sample}(1^\lambda, \mathbb{G}); (\alpha,\beta,U) \leftarrow \mathcal{A}(\mathbb{G}, G, \rho); \\ U \in \mathbb{G}; 0 \neq |\alpha| < 2^{\mathsf{poly}(\lambda)} \in \mathbb{Z}; |\beta| < 2^{\mathsf{poly}(\lambda)} \in \mathbb{Z}: \\ \beta U = \alpha G \wedge \frac{\beta}{\gcd(\alpha,\beta)} \neq e^k \text{ for } k \in \mathbb{N} \end{array}\right] \leq \mathsf{negl}(\lambda)$$

Definition A.1.10 (SI w.r.t. Sample). The **subgroup indistinguishability (SI) assumption w.r.t.** Sample holds for group $\mathbb{G}$ if for any PPT adversary $\mathcal{A}$, there is a negligible function negl, such that

$$\Pr\left[\begin{array}{c} (G,\rho) \leftarrow \mathsf{Sample}(1^\lambda, \mathbb{G}); H_0 \xleftarrow{\$} \mathbb{G}, H_1 \xleftarrow{\$} \langle G \rangle; \\ b \xleftarrow{\$} \{0,1\}, b' \leftarrow \mathcal{A}(\mathbb{G}, G, \rho, H_b): \\ b = b' \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

Definition A.1.11 (Hard (D,e,N)-relaxed DLOG-relation (w.r.t. Sample)). Let $\mathbb{G}$ be a group, $D,e,N \in \mathbb{N}$, and $\vec{G} = (G_0,\dots,G_N) \in \mathbb{G}^N$. The advantage $\mathsf{Adv}^{\text{rlx-dlog}}_{\mathbb{G},(D,e,N),\mathcal{A}}(\lambda)$ of $\mathcal{A}$ in the advantage against hardness of (D,e,N)-relaxed DLOG-relation **w.r.t.** Sample, is defined as the following probability:

$$\Pr\left[\begin{array}{c} \forall i = 1,\dots,N\colon (G_i,\rho_i) \leftarrow \mathsf{Sample}(1^\lambda, \mathbb{G}) \\ (C,d,m_0,\dots,m_N) \leftarrow \mathcal{A}(\mathbb{G}, G_0, \rho_0, \dots, G_N, \rho_N): \\ (C,d,m_0,\dots,m_N) \in \mathcal{R}_{D,e,N}(\vec{G}) \end{array}\right].$$

We say that finding (D,e,N)-relaxed DLOG-relations w.r.t. Sample is **hard**, if for every PPT adversary, there exists a negligible function negl such that $\mathsf{Adv}^{\text{rlx-dlog}}_{\mathbb{G},(D,e,N),\mathcal{A}}(\lambda) \leq \mathsf{negl}(\lambda)$.

(Note: We do not define hardness with subgroup setup, as this case does not occur.)

A.1.3. Proof of Lemma A.1.7

We give the full proof for Lemma A.1.7 below.

Proof. To item 1: This is immediate: If $\mathcal{A}$ outputs $(C,d,m_0,\dots,m_n)$, output $(C,d,m_0,\dots,m_n,0,\dots,0)$ to break (D,e,N)-relaxed DLOG-relation hardness with exactly the same success.

To item 2: For $N \leq 1$ this follows from the previous point. For $N \geq 2$, this follows with by borrowing randomization techniques from known prime order groups. Concretely, pick $\vec{r}_0, \vec{r}_1 \xleftarrow{\$} [0, 2N2^\lambda U_{\mathrm{up}}]^N$ and define the matrix

$$R = \begin{pmatrix} 1 & 0 \\ \vec{r}_0 & \vec{r}_1 \end{pmatrix} \in \mathbb{Z}^{(N+1)\times 2}$$

Let $(G'_0,\dots,G'_N)^\top = R(G_0,G_1)^\top$. The reduction hands $(G'_0,\dots,G'_N)$ to $\mathcal{A}$, which outputs $(d,C,m_0,\dots,m_N)$. The reduction then returns $(d,C,(m_0,\dots,m_N)R)$.

The success analysis will be information-theoretic. Let $K = \mathrm{ord}(G_0)$ be the order of the generated subgroup. Observe that information-theoretically $(\vec{r}_0,\vec{r}_1) \bmod K$ is almost uniform, namely the statistical distance to $\mathbb{Z}_K^N$ is at most $2^{-\lambda}$ to. Let g_1 be the DLOG of G_1 to G_0, i.e. $g_1 G_0 = G_1$. For simplicity, we now argue using the DLOGs, i.e. we argue over $\mathbb{Z}_K$ (mapping G_0 to 1 and G_1 to g_1), and we argue as though $\vec{r}_0$ and $\vec{r}_1$ are uniform modulo K. Observe that $R(1,g_1)^\top$ and $(R + \vec{v}(-g_1,1))(1,g_1)^\top$ have the same distribution for any $\vec{v} \in \mathbb{Z}_K^{N+1}$, that is

$$R(1,g_1)^\top \sim (R + \vec{v}(-g_1,1))(1,g_1)^\top \qquad \text{(A.1.1)}$$

In particular, this holds for uniformly random $\vec{v} \xleftarrow{\$} \mathbb{Z}_K^{N+1}$. Now consider $\mathcal{A}$'s output $(d, C, m_0, \dots, m_N)$. Let $c = \mathrm{dlog}_{G_0}(C)$, i.e $c \cdot G = C$, and let $\vec{m}^\top = (m_0, \dots, m_1)$. Then $dC = \sum_{i=0}^{m} m_i G_i = \vec{m}^\top R(G_0, G_1)^\top$ becomes $dc = \vec{m}^\top \vec{g}' = \vec{m}^\top R(1, g_1)$. Let $d' = d/\mathrm{gcd}(e^d, d)$, i.e. let $d' \mid d$ be the maximal factor of d which is coprime to e.

If $\mathcal{A}$ wins, then for some i we have $m_i/d \notin \mathbb{Z}[1/e]$. Moreover, following conditions are equivalent:

$$\begin{aligned} m_i/d \notin \mathbb{Z}[1/e] &\iff m_i/d' \notin \mathbb{Z}[1/e] \\ &\iff d' \nmid m_i \\ &\iff m_i \not\equiv_{d'} 0 \end{aligned}$$

Whenever $\vec{m}^\top R \not\equiv_{d'} 0$ holds, then $(d, C, \vec{m}^\top R)$ is a $(D, e, 1)$-relaxed DLOG-relation. Hence, we have to show that $\vec{m}^\top R \not\equiv_{d'} 0$ holds with high probability. From the equivalence of distributions in Eq. (A.1.1), we have

$$\begin{aligned} &\Pr[\vec{m}^\top R \equiv_{d'} \vec{0}] \\ &= \Pr[\vec{m}^\top (R + \vec{v}(-g_1, 1)) \equiv_{d'} \vec{0}] \\ &\leq \max_{\vec{m} \not\equiv_{d'} 0} \Pr[\vec{m}^\top \vec{v}(-g_1, 1) \equiv_{d'} \vec{\mu}] \end{aligned}$$

where the initial probabilities go over R, m, v, and we used for the inequality that we can maximize over $\vec{m}$ and R and let $\vec{\mu} = -\vec{m}^\top R$. Looking only at the second component of the equation $\vec{m}^\top \vec{v}(-g_1, 1) \equiv_{d'} \vec{\mu}$, namely, $\vec{m}^\top \vec{v} \equiv_{d'} \mu_2$, where $\vec{\mu} = (\mu_1, \mu_2)^\top$, suffices to upper-bound the probability.

Note that if $d' = 1$, the adversary loses, so w.l.o.g. $d' \neq 1$. Let $p^k \mid d'$ be a prime power dividing d' such that $k \in \mathbb{N}$ is minimal with:

- for all $i = 0, \dots, N$: $m_i \not\equiv_{p^k} 0$,
- for some $i = 0, \dots, N$: $m_i \equiv_{p^{k-1}} 0$.

If no such p exists, then $m_i \equiv_{d'} 0$ for all m_i, and again, $\mathcal{A}$ loses (because $m_i/d \in \mathbb{Z}[1/e]$ for all i). Thus, assume w.l.o.g. that such a prime p exists.

Now, we show
$$\Pr[\vec{m}^\top \vec{v} \equiv_{p^k} \mu_2] \leq 1/p.$$

For this, intuitively, we consider the p-adic digits and concentrate on the k-th digit, and show that $\vec{m}^\top \vec{v}$ has almost uniformly random k-th digit. To do so, first note that (by assumption) all m_i lie in $p^{k-1}\mathbb{Z}$, and for some i^*, we have $m_i \notin p^k\mathbb{Z}$. In other words, we can divide all m_i by p^{k-1}, and then one element, namely m_{i^*}/p^{k-1}, is not divisible by p anymore. Thus, after dividing and taking the equations modulo p, we see that m_{i^*}/p^{k-1} is invertible in $\mathbb{Z}_p$. (Note that $m_i/p^{k-1} \bmod p$ is exactly the k-th digit in basis p.) For the k-th digit of $\vec{m}^\top \vec{v}$, it is not hard to see that it is a uniformly linear combination of all the $m_i/p^{k-1} \bmod p$, since $\vec{v} \bmod p$ is uniform in $\mathbb{Z}_p^{N+1}$. But it is well-known that for $\vec{u} \xleftarrow{\$} \mathbb{Z}_p^{N+1}$ we have

$$\Pr[\vec{m}^\top \vec{u} \equiv_{p^k} \mu_2] = 1/p$$

and easy to check that $\Pr[\vec{a}^\top \vec{u} \equiv_p z] = 1/p$ if $\vec{u} \xleftarrow{\$} \mathbb{Z}_p^{N+1}$, $\vec{0} \neq \vec{a} \in \mathbb{Z}_p^{N+1}$, and $z \in \mathbb{Z}_p$.

Putting things together and accounting for the statistical distance of $2^{-\lambda}$ of the $\vec{r}_0$ and $\vec{r}_1$ from uniform over $\mathbb{Z}_K$, we have shown: If $\mathcal{A}$ does not lose, then we get

$$\Pr[\vec{m}^\top \vec{v} \neq \mu_2] \geq 1 - 1/p - 2^{-\lambda}$$

that is, with probability at least $1 - 1/p - \mathsf{negl}(\lambda)$ we find a non-trivial $(D, e, 1)$-relaxed relation. Thus, the claim follows.

To item 3: Observe that for $N = 0$, the $(D, e, 1)$-relaxed DLOG-relation specializes to hardness of finding (C, d, m) such that $dC = mG_0$ and $m/d \notin \mathbb{Z}[1/e]$, and $d \leq D$. Applying Remark A.1.4 with $C = W$, $d = \beta$ and $m = \alpha$, and noting that $m = \alpha = 0$ breaks neither assumption, we immediately obtain the claimed equivalence if $D = \infty$, and a one-sided implication otherwise.

To item 4: Since it makes no difference in the proof, we directly show that (D, e, N)-relaxed implies $(D, e, 0)$-relaxed DLOG-relation hardness, *if* $|\mathbb{G}|$ *has no prime factor smaller or equal to* D. We setup $G_i = \rho_i G_0$ for $\rho_i \xleftarrow{\$} [0, N2^\lambda U_{\mathrm{up}}^2 - 1]$, where U_{up} is an upper bound on the group order of $\mathbb{G}$. By Remark A.1.1, G_i is $1/N \cdot 2^{-\lambda} U_{\mathrm{up}}^{-1}$ close to a uniform element in $\langle G_0 \rangle$. By a union bound, $(G_1, \dots, G_N)$ is $2^{-\lambda} U_{\mathrm{up}}^{-1}$ close to uniform in $\mathbb{G}$. Let $n \leq U_{\mathrm{up}}$ be any number with $\gcd(n, \mathrm{ord}(G_0)) = 1$. Since G_i information-theoretically only reveals $\rho_i \bmod \mathrm{ord}(G_0)$, we see that $(\rho_1, \dots, \rho_N) \bmod n$ is $2^{-\lambda}$-close to uniform in $\mathbb{Z}_n$ (since $n \leq U_{\mathrm{up}}$). We apply this to $n = p^k/d$ later.

As in item 2, let (w.l.o.g.) $1 \neq d' \mid d$ be the maximal factor of d coprime to e, and let p be a prime and $k \in \mathbb{N}$ be minimal such that $m_i \equiv_{p^{k-1}} 0$ for all i, but $m_{i^*} \not\equiv_{p^k} 0$ for some $i^* \in \{0, \dots, N\}$. Such p and i^* exist since $d' \neq 1$.

Using the setup, we have $dC = m_0 G_0 + \sum_{i=1}^N m_i G_i = (m_0 + \sum_{i=1}^N \rho_i m_i) G_0$. Let $m := m_0 + \sum_{i=1}^N \rho_i m_i$. Observe that if $m \not\equiv_{p^k} 0$, then we have $dC = mG_0$ and $m \not\equiv_{d'} 0$, and hence $m/d \notin \mathbb{Z}[1/e]$ because d is not a power of e. Thus, we break e-fROOT. Now, we show that this happens with high probability.

Since we assumed that $m_{i^*} \not\equiv_{p^k} 0$, but $m_i \equiv_{p^{k-1}} 0$, we can argue almost as in item 2 that the linear combination $(\sum_{i=0}^N \rho_i m_i)/p^{k-1} \bmod p$ is zero with probability at most $1/p + \mathsf{negl}$. The only difference is that $\rho_0 = 1$. But it is easy to see that $z_0 + \sum_{i=1}^N u_i z_i \bmod p$ for arbitrary $z_i \in \mathbb{Z}$ and uniform $u_i \xleftarrow{\$} \mathbb{Z}_p$ is 0 with probability at most $1/p$, unless $z_i \bmod p = 0$ for all i.

Since $m := m_0 + \sum_{i=1}^N \rho_i m_i$, we find that $m \bmod p^k \neq 0$ with probability at least $1 - 1/p - \mathsf{negl}$, and hence, whenever $\mathcal{A}$ wins, the reduction wins with probability at least $1/3$. Thus, the claim follows. □

A.2. Further Remarks on Sharp's Soundness

The Sharp family satisfies correctness for short integers x in $\mathbb{Z}_p$, while it guarantees relaxed soundness, namely range membership of the rational representative $[x]_{\mathbb{Q}}$. Here, we discuss the behaviour of rational representatives and the soundness guarantees of (the variants of) Sharp from the perspective of possible use-cases.

A.2.1. Arithmetic Behaviour of $\mathbb{Q}_{M,D}$

Let $x_i, c \in \mathbb{Z}_p$ for $i \in [1, \ell]$. The usual integer representatives x_i behave very simply. Namely additions $x_1 + x_2$ and multiplications with constant $c \cdot x_1$ (and general multiplications $x_1 \cdot x_2$) work on representatives as long as it is ensured that no wraparound happens, i.e. the result is within $[-\frac{p-1}{2}, \frac{p-1}{2}]$. For example, for ℓ additions

$$\mathbb{Z} \ni \sum_{i=1}^{\ell} x_i = (\sum_{i=1}^{\ell} x_i) \bmod p \tag{A.2.1}$$

if $\ell M < p/2$ and each x_i is bound by M, everything works well. Similar claims hold for multiplications (with a constant $c \in \mathbb{Z}$). If there is a bound B on the total sum of values x_i, Eq. (A.2.1) holds if $B < p/2$ and independently of the total number of additions ℓ.

A.2.1.1. Overflow Conditions for $\mathbb{Q}_{M,D}$.

For rational representatives, overflows behaviour effectively boils down to computations with fractions. Let $x_i \in \mathbb{Z}_p$ and let $[x_i]_{\mathbb{Q}_{M,D}} = n_i/d_i \in \mathbb{Q}_{M,D}$ be the rational representative of x_i for $i \in [1, \ell]$. The sum $x_1 + x_2$ has rational representative $(n_1 d_2 + n_2 d_1)/(d_1 d_1) \in \mathbb{Q}_{2MD,D^2}$ and similarly for multiplications (with constants). Note that $\mathbb{Q}_{M,D}$ is "two-dimensional" in the sense that M and D are independent (but must satisfy $MD < p/2$), If $\mathbb{Q}_{M,D} \subseteq \mathbb{Q}_{M',D'}$ then representatives will coincide, but in general (e.g. if $M < M'$ but $D > D'$) $\mathbb{Q}_{M,D}$ and $\mathbb{Q}_{M',D'}$ representatives have no obvious relation. In analogy to the summation example (Eq. (A.2.1)), we can require $x_i \in \mathbb{Q}_{M,D}$ for $\ell MD^{2\ell-1} < p/2$ in order for

$$\sum_{i=1}^{\ell} [x_i]_{\mathbb{Q}_{M,D}} = [(\sum_{i=1}^{\ell} x_i) \bmod p]_{\mathbb{Q}_{M',D'}} \tag{A.2.2}$$

to hold in $\mathbb{Q}_{M',D'}$, where $M' \leq \ell MD^{\ell-1}$, $D' \leq D^{\ell}$. Similar claims hold for multiplication. Addition and multiplication of small integers $c \in \mathbb{Z}$ with $|c| < C$ behaves well: $c \cdot x_1 \in \mathbb{Q}_{CM,D}$.

Remark A.2.1. The potentially rapid growth of numerator and denominator under additions is one of the main sources of trouble when using rational representatives and relaxed soundness guarantees. Hence, they are less "friendly" in homomorphic operations on commitments.

A.2.1.2. Overflow Conditions in Hidden Order Groups.

With groups of hidden order, we show that the denominator is of the form $d_i = e^k \leq D$, cf. see Appendix A.3. In particular, a sum $x_1 + x_2$ now lies in $\mathbb{Q}_{2MD,D}$, i.e. the maximal possible denominator D is unchanged — we prevented the growth of D. The requirement for Eq. (A.2.2) to hold improves to ℓMD^2. When we further know a bound B on the sum of all numerators, the requirement becomes BD^2 and is independent of ℓ.

A.2.2. Remark on the Square Decomposition

We recall that the three square decomposition only shows relaxed range membership for fractions (unless one rounds to integers [CKLR21b] or has prior knowledge (Section 3.6.3)).

Lemma A.2.2 (Three Squares for Fractions). *Let $B \geq 1$, $x \in \mathbb{Q}$ and $\{x_i\}_{i=1}^{3} \in \mathbb{Q}$ such that $1 + 4x(B - x) = \sum_{i=1}^{3} x_i^2$. It holds that $x \in [-\frac{1}{4B}, B + \frac{1}{4B}]_{\mathbb{Q}}$.*

To show exact range membership for fractions, we can use the four square decomposition.

Lemma A.2.3 (Four Squares for Fractions). *Let $B \geq 1$, $x \in \mathbb{Q}$ and $\{x_i\}_{i=1..4} \in \mathbb{Q}$ and $B \in \mathbb{N}$. Further, let $x(B - x) = \sum_{i=1}^{4} x^2$. Then it holds that $x \in [0, B]_{\mathbb{Q}}$.*

Both decompositions can be calculated efficiently [RS86; PS19]. As the four square decomposition increases the communication (since we have to open an additional committed integer x_4), we present our range proofs using the three square decomposition. Replacing it with the four square decomposition leads to range proofs that guarantee exact range membership for the rational representative. Alternatively, if $\Gamma < 4B$ is ensured (e.g. by, if necessary, trading challenge size for repetitions), our soundness claims ensure that denominators $d \geq 4B$ violate soundness, i.e. $[0, B]_{\mathbb{Q}_{K',\Gamma}} = [-\frac{1}{4B}, B + \frac{1}{B}]_{\mathbb{Q}} \cap \mathbb{Q}_{K',\Gamma}$.

A.3. Augmented Soundness

We show how to "augment" a range proof with one hidden order group element in order to improve the soundness guarantee. The hidden order group can be instantiated with: (1) a class group for better additive homomorphic guarantees (cf. Appendix A.2.1.2) or (2) a RSA group for standard soundness with trusted setup.

A.3.1. Proof of Short Opening

Both the simple PoSO used in $\mathsf{Sharp}_{\mathsf{GS}}$ and the Batch-PoSO used in $\mathsf{Sharp}^{\mathsf{Po}}_{\mathsf{SO}}$ only ensure that the (committed) values are short as *fractions*, i.e. lie in $\mathbb{Q}_{M,D}$ for suitable M and D. It is easy to see, that these PoSOs also work over (commitments in) hidden order groups, as they are ignorant of the group order. Importantly, hidden order groups allow to mitigate the problems with homomorphic computations of fractions to some extent. Thus, we can achieve better soundness guarantees. Namely, under the hardness of (Γ, e)-relaxed DLOG relations, denominators d of an extracted witness $x = m/d$ must be of the form $d = e^k$, for $k \in \mathbb{N}_0$, i.e. the opening lies in $\mathbb{Z}[1/e]$ instead of $\mathbb{Q}$. In RSA groups, the hardness assumption is implied (with $e = 1$) by strong RSA, assuming safe primes are used, and therefore $x = m$, i.e. x is an *integer* representative. For class groups, the hardness assumption (with $e = 2$) is novel,[1] and see that $x = m/2^k$, i.e. a dyadic integer.

We leverage this in our range proof by adding an additional commitment to x in the hidden order group $\mathbb{H}$, and proving consistency and small opening. For RSA groups, we get $x \in \mathbb{Z}$ and hence *standard soundness.* For class groups, we get $x = \frac{m}{2^k}$. Despite allowing a denominator, this is a huge improvement over arbitrary rational representative, as homomorphic computation now pose a much smaller threat, since the committed values are forced to lie in $\frac{1}{2^{\log(\Gamma)}}\mathbb{Z}$, which is closed under addition (unlike $\mathbb{Q}_{\infty,D}$ for general D).

A.3.2. Augmented Range Proof

The modification to our schemes is surprisingly lightweight. It reuses the challenges of the standard scheme and does not require repetitions. The only additional communication is the commitment to x and the masked randomness required for the proof of short opening.

As additional setup, a Pedersen commitment key $ck_{\mathbb{H}} = (G_i', \rho_i)_{i=0}^{N}$, where $(G_i', \rho_i) \leftarrow \mathsf{Sample}(1^\lambda, \mathbb{H})$, with hiding parameter S is required. (We explicitly include the public coins ρ_i in the commitment key

[1] More precisely, we require a family of assumptions, which collapses to two assumptions when one assumes invertible sampling. Moreover, we show the assumptions are closely related to the better understood 2-fROOT and ORD assumptions. (See Lemma A.1.7) Unfortunately, our reductions do not apply for groups without invertible sampling. Thus, they cannot be used to provably justify security when using transparent setup. However, it is still a heuristic justification of their hardness.

due to the lack of invertible sampling in class groups, cf. Appendix A.1.2. Note that the correspondence of G'_i and ρ_i must be checked by the parties (once).) We mask values in $[0, S(\Gamma+1)^R]$ with masking algorithm mask'_r, masking randomness distribution R'_r, masking overhead L'_r and abort probability $\mathsf{p_r}'$. As stated above, the main difference is, that an additional MPed commitment C'_x to all x_i using $ck_{\mathbb{H}}$ is made (and sent by the prover), and knowledge of opening of C'_x is proven. We first describe the more complex case of $\mathsf{Sharp_{GS}}$.

A.3.2.1. Necessary Modifications to $\mathsf{Sharp_{GS}}$.

We describe only the modifications of Algorithm 1 ($\mathsf{Sharp_{GS}}$) below, using the same variable names as in $\mathsf{Sharp_{GS}}$.

- The prover's first flow (e.g. after Line 12) is changed as follows: Additionally commit the x_i in $\mathbb{H}$.
  1. $C'_x = r'_x G'_0 + \sum_{i=1}^N x_i G'_i$, where $r'_x \xleftarrow{\$} [0, S]$.

  Now, compute the first message of the proof of short opening in $\mathbb{H}$.

  2. Set $\widetilde{r}'_x \xleftarrow{\$} \mathsf{R}'_r$ and let $\widetilde{x}_{k,i}$ be as in $\mathsf{Sharp_{GS}}$.
  3. Let $\widetilde{x}'_i = \sum_{k=1}^R (\Gamma+1)^{k-1} \widetilde{x}_{k,i}$
  4. Set $D'_x = \widetilde{r}'_x G'_0 + \sum_{i=1}^N \widetilde{x}'_i G'_i$

  Modify the sent message as follows:

  1. Add C'_x to the message.
  2. With the hash optimization, add D'_x to the list of hashed messages. (Without it, add D'_x to the message.)
- The verifier's challenge is unmodified. Recall that $\gamma_k \in [0, \Gamma]$ for $k \in [1, R]$ are the challenges.
- The provers's response (e.g. after Line 18) is changed as follows: Compute the "synthesized challenge" γ'.
  1. Set $\gamma' = \sum_{k=1}^R \gamma_k (\Gamma+1)^{k-1} \in [0, (\Gamma+1)^R - 1]$.

  Compute the masked opening randomness.

  2. Set $t'_x = \mathsf{mask}'_r(\gamma' \cdot r'_x, \widetilde{r}'_x)$.

  Abort if any masking failed. Modify the sent message as follows: Add t' to the message.
- The verifier's check (e.g. after Line 8) is changed as follows. Let the "synthesized" challenge and responses be:
  1. $\gamma' = \sum_{k=1}^R \gamma_k (\Gamma+1)^{k-1} \in [0, (\Gamma+1)^R - 1]$
  2. $z'_i = \sum_{k=1}^R (\Gamma+1)^{k-1} \cdot z_{k,i}$

  Add the following computations and checks:

  3. Compute the "synthesized" γ' and z'_i for $i \in [1, N]$.
  4. Compute $F'_x = -\gamma' C'_x + t'_x \cdot G'_0 + \sum_{i=1}^N z'_i \cdot G'_i$
  5. With the hash optimization, modify the check of Δ by including F'_x in the list of messages. (Without it, check $F'_x = D_x$.)

A.3.2.2. Necessary Modifications to $\mathsf{Sharp}^{\mathsf{Po}}_{\mathsf{SO}}$

For $\mathsf{Sharp}^{\mathsf{Po}}_{\mathsf{SO}}$, the analogous changes of $\mathsf{Sharp}_{\mathsf{GS}}$ are applied to Phase 2. Since there are no repetitions in Phase 2, no "synthesized" γ' needed, hence $\gamma' = \gamma$ and $[0, \widehat{\Gamma}]$ is used as challenge space. We stress, that maskings which are shared with $\mathbb{H}$ (concretely, $z_i = \mathsf{mask}_x(\gamma x_i, \widetilde{x}_i)$) must now be computed over $\mathbb{Z}$ (and not modulo p, since the group orders of $\mathbb{H}$ and $\mathbb{G}_{\mathsf{com}}$ are "incompatible").

A.3.2.3. Efficiency

We consider the schemes with hash optimization applied. Then, compared to $\mathsf{Sharp}_{\mathsf{GS}}$ (resp. $\mathsf{Sharp}^{\mathsf{Po}}_{\mathsf{SO}}$), the additional communication is a single element in $\mathbb{H}$ (namely C'_x), and the integer $t'_x \in \mathsf{R}'_r$. Additional computation for the prover's is computing C'_x and D'_x. For the verifier, it is the computation of F'_x. Other changes are negligible.

A.3.2.4. Security

$\mathsf{Sharp}^{+\mathsf{HO}}_{\mathsf{GS}}$ is correct, non-abort SHVZK and provides a *strengthened* relaxed soundness guarantee. Informally, the committed x_i are guaranteed to have rational representatives in $[-\frac{1}{4B}, B + \frac{1}{4B}]_{\mathbb{Q}}$, where due to hardness of (Γ, e, N)-relaxed DLOG-relations in $\mathbb{H}$, x_i is of the form m/e^{ℓ} for $m \in [-2(B\Gamma + 1)L, 2(B\Gamma + 1)L]$, $\ell \le \log(\Gamma)$. To deal with the lack of invertible sampling in the class group setting, we consider an explicit sampling algorithm Sample for uniform group elements, and hardness of assumptions w.r.t. Sample.

Theorem A.3.1. *Let* Sample *be a sampling algorithm for* $\mathbb{G}$*. The scheme* $\mathsf{Sharp}^{+\mathsf{HO}}_{\mathsf{GS}}$ *has correctness error at most* $1 - (1 - \mathsf{p_r}')^N[(1 - \mathsf{p_r})^3 \cdot (1 - \mathsf{p_x})^N]^R$*. It is non-abort SHVZK under the SEI assumptions on* $\mathbb{G}$ *and the SEI and the SI assumptions on* $\mathbb{H}$*. Is has relaxed soundness for the relation*

$$\begin{aligned}\mathcal{R}_{\mathsf{Ext}} = \Big\{ (x_1, \dots, x_N, r) \colon & C_x = r_x G_0 + \sum_{i=1}^{N} x_i G_i \\ & \wedge \exists m_i \in \mathbb{Z}, k \in \mathbb{N}_0 \colon -\frac{1}{4B} \le \frac{m_i}{e^k} \le B + \frac{1}{4B} \\ & \wedge x_i \equiv_q \frac{m_i}{e^k} \wedge |m_i| \le (B\Gamma + 1)L_x \wedge 1 \le e^k \le \Gamma \Big\}.\end{aligned}$$

under the DLOG, SEI assumptions on $\mathbb{G}$*, and the DLOG, SEI, SI, assumption and hardness of* (Γ, e)*-relaxed DLOG-relations in* $\mathbb{H}$*, where all asumptions are all w.r.t. to* Sample*. The knowledge error is* $(\frac{2}{\Gamma+1})^R$*. Concretely, with the hash-optimization, we have following reductions:*

- *For every adversary* $\mathcal{A}$ *against non-abort SHVZK, there are adversaries* $\mathcal{B}_{\mathbb{G},SEI}$, $\mathcal{B}_{\mathbb{H},SEI}$, $\mathcal{B}_{\mathbb{H},SI}$ *whose runtime is roughly that of* $\mathcal{A}$ *and so that* $\mathsf{Adv}^{\text{na-hvzk}}_{\mathcal{A}} \le \mathsf{Adv}^{\text{sei}}_{\mathbb{H},\mathcal{B}_{\mathbb{G}_{\mathsf{com}},SEI}} + \mathsf{Adv}^{\text{sei}}_{\mathbb{H},\mathcal{B}_{\mathbb{H},SEI}} + \mathsf{Adv}^{\text{si}}_{\mathbb{H},\mathcal{B}_{\mathbb{H},SI}}$.
- *For every adversary* $\mathcal{A}$ *against knowledge which runs at most* T *steps, there are adversaries* $\mathcal{B}_{CR}$, $\mathcal{B}_{\mathbb{G},DLOG}$, $\mathcal{B}_{\mathbb{H},DLOG}$, $\mathcal{B}_{\mathbb{H},SI}$, $\mathcal{B}_{\mathbb{H},rlxDLOG}$, *whose* expected *runtime is roughly* $3 \cdot R \cdot T$*, and so that* $\mathsf{Adv}^{\text{ke}}_{\mathcal{A}} \le (\frac{2}{\Gamma+1})^R + \mathsf{Adv}^{\text{crhf}}_{\mathsf{Hash},\mathcal{B}_{CR}} + \mathsf{Adv}^{\text{dlog}}_{\mathbb{G},\mathcal{B}_{\mathbb{G},DLOG}} + \mathsf{Adv}^{\text{dlog}}_{\mathbb{H},\mathcal{B}_{\mathbb{H},DLOG}} + \mathsf{Adv}^{\text{rlx-dlog}}_{\mathbb{H},(\Gamma,e,N),\mathcal{B}_{\mathbb{H},rlxDLOG}}$.

To be precise, we consider the S*-bounded SEI assumption in* $\mathbb{G}$ *and the* S*-bounded SEI assumption in* $\mathbb{H}$*.*

The analogous adaption of Theorem A.5.1 holds for $\mathsf{Sharp}^{+\mathsf{HO}}_{\mathsf{PoSO}}$*, where the same additional terms for reductions in* $\mathbb{H}$ *appear.*

Correctness follows by inspection. The soundness follows essentially as for the unmodified $\mathrm{Sharp}_{\mathrm{GS}}$ (Theorem A.5.1), except that hardness of (Γ, e)-relaxed DLOG-relations is used to additionally argue that $x_i \in \mathbb{Z}[1/e]$ as sketched in Appendix A.3.1. Zero-knowledge follows almost exactly as for $\mathrm{Sharp}_{\mathrm{GS}}$. The full proof is in Appendix A.5.3.

A.4. Proofs for Shortness Testing

A.4.1. Proof of Lemma 3.3.7

Lemma 3.3.7 (Regular Spacing of $\mathbb{S}_d$). *Suppose* $1 < d < p$ *and* $\gcd(d,p) = 1$ *and consider the set*

$$\mathbb{S}_d \equiv_p \{\frac{i}{d} \bmod p \mid i \in [0,\dots,d-1]\} \subseteq \mathbb{Z}_p. \tag{3.3.5}$$

Then $\mathbb{S}_d = \{\lceil ip/d \rceil \mid i \in [0,\dots,d-1]\}$ *and the minimal distance* $\delta = \min_{x \neq y \in \mathbb{S}_d} |x-y|$ *satisfies* $\delta = \lfloor \frac{p}{d} \rfloor$.

Proof. Define $s_i := \lceil \frac{ip}{d} \rceil$ for $i = 0,\dots,d-1$. Observe that, by Eq. (3.3.3),

$$s_i = \left\lceil \frac{ip}{d} \right\rceil = \frac{ip}{d} + \frac{ip \bmod d}{d}$$

and after multiplication by d,

$$ds_i = ip + (ip \bmod d).$$

This equality holds over $\mathbb{Z}$. Modulo p, we find $ds_i \equiv_p (ip \bmod d)$. Since $\gcd(d,p) = 1$, all $ip \bmod d$ are distinct, hence $[0, d-1] = \mathbb{Z}_d = \{ip \bmod d \mid i \in [0, d-1]\}$. Dividing by d (over $\mathbb{Z}_p$), we find that

$$\{s_0,\dots,s_{d-1}\} \equiv_p \{j/d \in \mathbb{Z}_p \mid j \in [0, d-1]\}$$

Thus, we have shown that the set $\mathbb{S}_d$ indeed consists of the $s_i = \lceil \frac{ip}{d} \rceil$. The closest elements to s_i are $s_{(i+1 \bmod d)}$ or $s_{(i-1 \bmod d)}$, and (since the space is "circular") it suffices to consider the distances $s_{(i+1 \bmod d)} - s_{(i \bmod d)}$ to lower-bound the minimal distance δ in $\mathbb{S}_d$. For $i = 0,\dots,d-2$, we find

$$\left\lfloor \frac{p}{d} \right\rfloor \leq \left\lceil \frac{(i+1)p}{d} \right\rceil - \left\lceil \frac{ip}{d} \right\rceil = s_{i+1} - s_i \leq \left\lceil \frac{p}{d} \right\rceil$$

as claimed. For $i = d-1$, we find $s_0 - s_{d-1} \equiv_p p - s_{d-1}$, and since $p = \left\lceil \frac{dp}{d} \right\rceil$, the claim follows as above. In fact, $p - \lceil \frac{(d-1)p}{d} \rceil = \lfloor \frac{p}{d} \rfloor$, since $\lceil \frac{(d-1)p}{d} \rceil + \lceil \frac{p}{d} \rceil = p + 1$ (since $d \nmid p$). □

A.4.2. Proof of Lemma 3.3.8

Lemma 3.3.8. *Suppose* $d \in \mathbb{N}$ *and* $\gcd(d,p) = 1$ *and* $u \xleftarrow{\$} [0,\dots,d-1]$. *Let* $\mu, K \in \mathbb{N}$ *be arbitrary. Then for* $1 < d < p$ *we have*

$$\Pr\left[\frac{u}{d} \in_{\mathbb{Z}_p} [0,K]_{\mathbb{Z}_p} + \mu\right] \leq \frac{1}{d}\left\lceil \frac{K+1}{\lfloor \frac{p}{d} \rfloor} \right\rceil \tag{3.3.6}$$

and for $d > p$, *we have*

$$\Pr\left[\frac{u}{d} \in_{\mathbb{Z}_p} [0,K]_{\mathbb{Z}_p} + \mu\right] \leq 2\frac{K+1}{p} \tag{3.3.7}$$

where the probability is over u. Combining the conditions gives

$$\Pr\Big[\frac{u}{d} \in_{\mathbb{Z}_p} [0,K]_{\mathbb{Z}_p} + \mu\Big] \leq \frac{1}{d} + 2\frac{K+1}{p} \tag{3.3.8}$$

Proof. First we show Eq. (3.3.6). By Lemma 3.3.7, the distance between points in $\mathbb{S}_d = \{\frac{i}{d} \bmod p \mid i \in [0,\dots,d-1]\}$ is at least $\delta = \lfloor p/d \rfloor$. Consequently, at most $\lceil \frac{K+1}{\delta} \rceil$ points can lie in an interval with $K+1$ elements, e.g. $[0,K]_{\mathbb{Z}_p} + \mu$, by a simple counting argument.

The next claim, Eq. (3.3.7) follows by a simple direct analysis. Namely, $u \bmod p$ is distributed almost uniformly over $[0,p-1]$, in particular $\rho_{\sup}(u/U_{\mathbb{Z}_p}) \leq 2$. Moreover, multiplication with $1/d$ is a bijection modulo p since $\gcd(d,p) = 1$, so $U_{\mathbb{Z}_p}/d \bmod p$ is distributed as $U_{\mathbb{Z}_p}$. Consequently, $\Pr[u/d \in_{\mathbb{Z}_p} [0,K]_{\mathbb{Z}_p}] \leq \rho_{\sup}(u/U_{\mathbb{Z}_p}) \cdot \Pr[U_{\mathbb{Z}_p} \in_{\mathbb{Z}_p} [0,K]_{\mathbb{Z}_p}] \leq 2\frac{K+1}{p}$.

Finally, Eq. (3.3.8) follows by case distinction. Let $\widehat{K} := K+1$. For $d > p$, it follows immediately from Eq. (3.3.7). For $d < p/2$, we get

$$\left\lfloor \frac{\widehat{K}}{\delta} \right\rfloor = \left\lfloor \frac{\widehat{K}}{\lfloor \frac{p}{d} \rfloor} \right\rfloor \leq \frac{\widehat{K}}{\lfloor \frac{p}{d} \rfloor} \leq \frac{\widehat{K}}{p/d-1} \leq d \cdot \frac{\widehat{K}}{p-d} \leq d \cdot \frac{2(K+1)}{p}$$

since $p/d-1 = (p-d)/d$ and $p-d \geq p/2$. Using $\lceil \widehat{K}/\delta \rceil \leq 1 + \lfloor \widehat{K}/\delta \rfloor$ it follows that $\frac{1}{d}\lceil \widehat{K}/\delta \rceil \leq \frac{1}{d}(1+2d\frac{K+1}{p})$. For $p/2 < d < p$, we have $\delta = \lfloor p/d \rfloor = 1$ and $1/d < 2/p$, hence $\frac{1}{d}\lceil \widehat{K}/\delta \rceil = \frac{\widehat{K}}{d} \leq 2\frac{K+1}{p}$. □

A.4.3. Proof of Lemma 3.3.9

Lemma 3.3.9. *Let $p,d,a,b,\mu,D,K \in \mathbb{N}$ and suppose $u \xleftarrow{\$} [0,D]$ is a uniform random variable. Let $\mathbb{S}_d \equiv_p \{i/d \mid i \in \mathbb{Z}_d\} \subseteq \mathbb{Z}_p$ as usual, and likewise $\mathbb{S}_b$. Suppose that* $\gcd(d,p) = 1$, *and $b \mid d$, and that*

$$b(K+1) + Da < \left\lfloor \frac{p}{d/b} \right\rfloor. \tag{3.3.9}$$

Then we have

$$\sum_{s \in \mathbb{S}_d} \Pr\Big[u\frac{a}{b} \in_{\mathbb{Z}_p} [0,K]_{\mathbb{Z}_p} + \mu + s\Big] \leq \left\lceil \frac{b(K+1)}{a} \right\rceil \frac{1}{D+1}. \tag{3.3.10}$$

Proof. Consider

$$\begin{aligned}
&\Pr\big[u\tfrac{a}{b} \in_{\mathbb{Z}_p} [0,K]_{\mathbb{Z}_p} + \mu + \mathbb{S}_d\big] \\
\leq &\Pr\big[ua \in_{\mathbb{Z}_p} [0,bK]_{\mathbb{Z}_p} + \mu' + b \cdot \mathbb{S}_d\big] \\
\leq &\Pr\big[ua \in_{\mathbb{Z}_p} [0,bK]_{\mathbb{Z}_p} + \mu' + \mathbb{S}_{d/b} + [0,b-1]\big] \\
\leq &\Pr\big[ua \in_{\mathbb{Z}_p} [0,b(K+1)-1]_{\mathbb{Z}_p} + \mu' + \mathbb{S}_{d/b}\big]
\end{aligned}$$

where we used that $\mu' = b\mu$, $b \cdot [0,K] \subseteq [0,bK]$, and $b \cdot \mathbb{S}_d \subseteq \mathbb{S}_{d/b} + [0,b-1]$. (The latter follows since $b \cdot i/d = i/d'$ where $d' = d/b$, and $i < d$, so $i/d' = (i \bmod d')/d' + \lfloor i/d' \rfloor \in \mathbb{S}_{d/b} + [0,b-1]$.) For brevity, define $K' = b(K+1)-1$.

Claim A.4.1. *If $u'a \in_{\mathbb{Z}_p} [0,K']_{\mathbb{Z}_p} + \mu' + s'$ for some choice $u' \in [0,D]$ and $s' \in \mathbb{S}_{d/b}$, then s' is unique, i.e. there exists no other choice $u'' \in [0,D]$, $s'' \in \mathbb{S}_{d/b}$ with $s' \neq s''$ and $u''a \in_{\mathbb{Z}_p} [0,K']_{\mathbb{Z}_p} + \mu' + s''$.*

Proof. Suppose otherwise. Observe that $ua \in [0, Da]$. Hence the distance of $u''a$ and $u'a$ is at most Da, Considering the "slack" of $[0, K']$, the points $s', s'' \in \mathbb{S}_{d/b}$ can therefore be as most $K' + Da$ far apart. The minimal distance in $\mathbb{S}_{d/b}$ is $\lfloor p/(d/b) \rfloor$. However, by assumption (Eq. (3.3.9)) $K' + Da < \lfloor p/(d/b) \rfloor$. Thus, $s' \neq s''$ must be too far from each other, which is a contradiction. □

We have just shown that there is a at most one $s' \in \mathbb{S}_{d/b}$ for which $u'a \in_{\mathbb{Z}_p} [0, K']_{\mathbb{Z}_p} + \mu' + s'$ can happen. Thus, we find

$$\Pr\big[ua \in_{\mathbb{Z}_p} [0, K']_{\mathbb{Z}_p} + \mu' + \mathbb{S}_{d/b}\big] \leq \Pr\big[ua \in_{\mathbb{Z}_p} [0, K']_{\mathbb{Z}_p} + \mu''\big]$$

where $\mu'' = \mu' + s'$. But it is clear that at most $\lceil (K'+1)/a \rceil$ choices of u can lie in an interval with $K'+1$ elements (since $Da < p$). With $K' + 1 = b(K+1)$, it follows that

$$\Pr\big[ua \in_{\mathbb{Z}_p} [0, K']_{\mathbb{Z}_p} + \mu''\big] \leq \left\lceil \frac{b(K+1)}{a} \right\rceil \frac{1}{D+1}.$$

□

A.4.4. Proof of Lemma 3.3.11

Lemma A.4.2. *Suppose $1 \neq d \in \mathbb{N}$ and let u_i be random variables in $\mathbb{Z}_d = [0, \ldots, d-1]$ for $i = 1, \ldots, N$. Fix some arbitrary $a_i \in [0, d-1]$ with $\operatorname{lcm}(a_1, \ldots, a_N) = d$. Define*

$$F \colon \mathbb{Z}_d \longrightarrow \mathbb{Z}_d, \qquad F(u_1, \ldots, u_N) = \sum_{i=1}^{N} u_i \cdot a_i \mod d \tag{A.4.1}$$

There exist $q_1, \ldots, q_N \in \mathbb{N}$ such that

1. *All q_i are coprime.*
2. *$q_i \mid \operatorname{ord}_{\mathbb{Z}_d}(a_i)$.*
3. *$\prod_{i=1}^{N} q_i = d$.*

Define $Z = \prod_{i=1}^{N} \mathbb{Z}_{q_i}$ and following homomorphisms:

- *The projections $\pi_i \colon \mathbb{Z}_d \longrightarrow \mathbb{Z}_{q_i}$ and the CRT map $\pi \colon \mathbb{Z}_d \longrightarrow Z$ with $\pi(x) = (\pi_1(x), \ldots, \pi_N(x))$.*
- *The injections $\iota_i \colon \mathbb{Z}_{q_i} \longrightarrow \mathbb{Z}_d$ defined by $x \mapsto \alpha_i \cdot x \mod d$, where $\alpha_i := \frac{d}{q_i} \cdot ((\frac{d}{q_i})^{-1} \mod q_i) \in \mathbb{Z}$, and the combined injections $\iota \colon Z \longrightarrow \mathbb{Z}_d^N$ as $\iota((x_1, \ldots x_N)) = (\iota_1(x_1), \ldots, \iota_N(x_N))$.*
- *The CRT map $\phi \colon Z \longrightarrow \mathbb{Z}_d$, $\phi((x_1, \ldots, x_N)) = \sum_{i=1}^{N} \iota_i(x_i) = \sum_{i=1}^{N} \alpha_i x_i$.*

Recall that π and ϕ are the bijections of the Chinese remainder theorem (CRT).

With this, we have:

4. *Restricted to $\iota(Z)$, the map $f \colon \iota(Z) \longrightarrow \mathbb{Z}_d$, $f = F|_{\iota(Z)}$ is an isomorphism.*
5. *For uniform $(v_1, \ldots, v_N) \xleftarrow{\$} Z$, we find that*

$$F(\iota(v_1, \ldots, v_N)) = f(v_1, \ldots, v_N) = \sum_{i=1}^{N} \iota_i(v_i) \cdot a_i \mod d$$

is uniform in $\mathbb{Z}_d$. Consequently, for uniform $(u_1, \ldots, u_N) \xleftarrow{\$} \iota(Z) \leq \mathbb{Z}_d^N$, also $F(u_1, \ldots, u_N)$ is uniform in $\mathbb{Z}_d$.

The lemma is essentially an application of the Chinese remainder theorem (CRT) and some standard computations. We provide a small example: Suppose $N = 2$ and $d = 300 = 2^2 \cdot 3 \cdot 5^2$, $a_1 = 15$ and $a_2 = 4$. Then $\operatorname{ord}_{\mathbb{Z}_d}(a_1) = 300/15 = 2^2 \cdot 5$ and $\operatorname{ord}_{\mathbb{Z}_d}(a_2) = 300/20 = 3 \cdot 5^2$. Thus, let $q_1 = 2^2$ and $q_2 = 3 \cdot 5^2$ (i.e. gather the largest prime powers in the q_i's). Clearly, $\mathbb{Z}_{300} \cong \mathbb{Z}_4 \times \mathbb{Z}_{75}$ by the CRT. It's easy to check the claims as well; they are almost directly implied by the CRT.

Proof. First, we show the existence of q_i's as claimed and some resulting properties. For this, let $h_i = \operatorname{ord}_{\mathbb{Z}_d}(a_i)$. Let $h_i = p_1^{e_{i,1}} \cdot \ldots p_r^{e_{i,r}}$ for distinct primes p_j and exponents $e_{i,j} \in \mathbb{N}_0$. Define q_1 as the product of those $p_j^{e_{1,j}}$ where $e_{1,j}$ is the maximal exponent (over all $j = 1, \ldots r$). The other q_i are defined analogously. If for fixed fixed j, there are multiple i such that h_i has the *maximal* exponent $e_{i,j}$ for p_j, then $p_j^{e_{i,j}}$ is part of (only!) the q_i with the smallest index i. By construction, $q_1, \ldots, q_N$ satisfy the required properties.

Conversely, the required properties enforce this structure, up to choices where for fixed j, multiple indices i have the maximal prime power $p_i^{e_{i,j}}$. This essentially follows from q_is being coprime, hence each prime (power) appears in at most one q_i, and $\prod_i^d q_i = d$, hence each prime (power) appears in at least one q_i.

Lastly, note that from the abstract properties, we get $(a_i \bmod q_i) \in \mathbb{Z}_{q_i}^\times$. This can be seen via the CRT: We have $\mathbb{Z}_d = \mathbb{Z}_{d/q_i} \times \mathbb{Z}_{q_i}$ via the CRT, since $\gcd(q_i, d/q_i) = 1$ by definition of q_i. Moreover, projecting a_i to $\mathbb{Z}_{q_i}$, a_i is must be generator of $\mathbb{Z}_{q_i}$ (or $q_i \nmid \operatorname{ord}_{\mathbb{Z}_d}(a_i)$, a contradiction).

Now we turn to Item 4. By the CRT, we have $Z = \prod_{i=1}^N \mathbb{Z}_{q_i} \cong \mathbb{Z}_{\prod_i q_i} = \mathbb{Z}_d$, and the isomorphism connecting Z and $\mathbb{Z}_d$ are π and ϕ. Moreover,

$$F(\iota(v_1, \ldots, v_N)) = \sum_{i=1}^N \iota_i(v_i) \cdot a_i = \sum_{i=1}^N \alpha_i v_i \cdot a_i = \sum_{i=1}^N v_i \cdot \frac{d}{q_i} a_i'$$

where $a_i' = a_i((\frac{d}{q_i})^{-1} \bmod q_i)$ by definition of α_i. The order of $\frac{d}{q_i} a_i' = \alpha_i a_i$ in $\mathbb{Z}_d$ is exactly q_i (since α_i is invertible modulo q_i). As $\mathbb{Z}_d$ is cyclic, each subgroup is uniquely identified by its order, and we conclude that the image $F(\iota(0, \ldots, \mathbb{Z}_{q_i}, 0, \ldots))$ is *the* subgroup of order q_i in $\mathbb{Z}_d$. Since $\prod_{i=1}^N q_i = d$, these subgroups span $\mathbb{Z}_d$ (again, by the CRT) and therefore f is surjective (and hence, bijective since $|Z| = |\mathbb{Z}_d|$).

Lastly, item 5 follows immediately from $f \colon Z \to \mathbb{Z}_d$ being an isomorphism, so in particular a bijection. □

A.4.5. Proof of Lemma 3.3.12

Lemma 3.3.12 (Core Lemma). *Let $D, M \in \mathbb{N}$ and suppose $2DM < p$. Let $x_i = \frac{m_i}{d_i}$ where $d_i \in [1, D]$ and $m_i \in [-M, M]$ for $i = 1, \ldots, N$. Let $\gamma_i \xleftarrow{\$} [0, D]$. Define*

$$S = \sum_{i=1}^N \gamma_i \cdot \frac{m_i}{d_i} \bmod p \tag{3.3.11}$$

Let $I \subseteq [1, N]$ denote the set of indices which minimizes $d := \operatorname{lcm}(\{d_i \mid i \in I\})$ under the constraint that $d > D$, or $I = [1, N]$ if $\operatorname{lcm}(d_1, \ldots, d_N) \le D$. Let $K \in \mathbb{N}$, let $\beta = \min(|I|, \operatorname{primlmin}(D + 1))$, and let $K' := K + 2\beta M$. Then, for arbitrary $\mu \in \mathbb{Z}_p$, we have

$$\Pr\big[S \in [0, K]_{\mathbb{Z}_p} + \mu\big] \le 4 \cdot \begin{cases} \frac{1}{d} & \text{if } d(K' + 1) < p \\ \frac{1}{d} + 2\frac{K'+1}{p} & \text{always} \end{cases} \tag{3.3.12}$$

Now, suppose additionally that $d \leq D$ and $D(K' + DM + 2) < p$. If $\frac{d}{d_i}|m_i| > K'$ for some $i \in [1, N]$, then

$$\Pr\big[S \in [0,K]_{\mathbb{Z}_p} + \mu\big] \leq \frac{8}{D+1}. \tag{3.3.13}$$

Proof. We assume w.l.o.g. that $D \geq 2$, $K \geq 1$, $d > 1$ and $N > 1$; the excluded cases are straightforward. As a first step, we impose conditions on I, N and d.

Claim A.4.3. *We can w.l.o.g. assume that $I = \{1, \ldots, N\}$ and that for any subset I' of I, $\operatorname{lcm}(d_{i'} \mid i' \in I) < \operatorname{lcm}(d_i \mid i \in I)$, that is, I is a minimal subset w.r.t. the least common multiple $d = \operatorname{lcm}(d_i \mid i \in I)$.*

Proof. First of all, we show that w.l.o.g. $I = \{1, \ldots, N\}$. For this, note that

$$\sum_{i=1}^{N} \gamma_i \frac{m_i}{d_i} \in_{\mathbb{Z}_p} [0,K]_{\mathbb{Z}_p} + \mu \iff \sum_{i \in I} \gamma_i \frac{m_i}{d_i} \in_{\mathbb{Z}_p} [0,K]_{\mathbb{Z}_p} + \underbrace{(\mu - \sum_{i \notin I} \gamma_i \frac{m_i}{d_i})}_{\mu'}.$$

Thus, if the lemma holds only for the "partial-sum" $I \subseteq \{1, \ldots, N\}$, it follows for the "complete sum" over $\{1, \ldots, N\}$, by conditional probability and using that the bounds in Eq. (3.3.12) must hold for *arbitrary μ*, in particular μ' (by induction over the instance size N). Thus, w.l.o.g. $I = \{1, \ldots, N\}$.

Now, suppose removing some d_i, w.l.o.g. d_N, does not affect the least common multiple, i.e. $\operatorname{lcm}(d_{i'} \mid i' \in I \setminus \{N\}) = d$. Then instead of I we could use $I' = I \setminus \{N\}$. By the above, w.l.o.g. we can assume $I' = \{1, \ldots, N\}$ again. Overall, we can assume "minimality" of I and $N = |I|$. □

Since w.l.o.g. $I = \{1, \ldots, N\}$, from now on we will mostly ignore the index set I.

Before diving into the proof, recall that

$$\frac{a}{b} = \frac{a \bmod b}{b} + \left\lfloor \frac{a}{b} \right\rfloor. \tag{A.4.2}$$

Let $d = \operatorname{lcm}(\{d_i \mid i \in I\})$ as in the claim. To motivate our approach, we first rewrite Eq. (3.3.11) with common denominator d and apply Eq. (A.4.2) to find

$$\begin{aligned}
&\Pr\big[S \in [0,K]_{\mathbb{Z}_p} + \mu\big] \\
&= \Pr\left[\sum_{i=1}^{N} \gamma_i \cdot \frac{m_i}{d_i} \in_{\mathbb{Z}_p} [0,K]_{\mathbb{Z}_p} + \mu\right] \\
&= \Pr\left[\frac{1}{d}\left(\sum_{i=1}^{N} \gamma_i \cdot m_i \frac{d}{d_i}\right) \in_{\mathbb{Z}_p} [0,K]_{\mathbb{Z}_p} + \mu\right] \\
&= \Pr\left[\frac{1}{d}\left(\left(\sum_{i=1}^{N} \gamma_i \cdot m_i \frac{d}{d_i}\right) \bmod d\right) + \left\lfloor \sum_{i=1}^{N} \gamma_i m_i \frac{1}{d_i} \right\rfloor \in_{\mathbb{Z}_p} [0,K]_{\mathbb{Z}_p} + \mu\right]
\end{aligned}$$

Observe that we now have a sum modulo d and are *almost* in the situation of Lemma A.4.2, which in turn would allow us to apply Lemma 3.3.8. But $\sum_{i=1}^{N} \gamma_i \cdot m_i \frac{d}{d_i} \bmod d$ need not be uniform modulo d, and $\lfloor \sum_{i=1}^{N} \gamma_i m_i \frac{1}{d_i} \rfloor$ is a stochastically dependent "error term". Thus, we will change the distribution of the γ_i in a suitable manner to obtain two independent sums.

For better tightness, we use the distribution suggested by Lemma A.4.2. Let q_i be as in Lemma A.4.2. Together with Claim A.4.3, we get the following properties.

Claim A.4.4. *We have that all q_i are coprime, $q_i > 1$ for all i, $\prod_{i=1}^{N} q_i = d$, $q_i \mid \operatorname{ord}_{\mathbb{Z}_d}(m_i d/d_i) \mid d_i \mid d$, and $N \leq \mathsf{primlmin}(D+1)$.*

Proof. Directly from Lemma A.4.2, we see that all q_i are coprime, $q_i \mid \operatorname{ord}_{\mathbb{Z}_d}(m_i d/d_i)$, and $\prod_{i=1}^{N} q_i = d$. The divisibility chain is completed via $\operatorname{ord}_{\mathbb{Z}_d}(m_i d/d_i) \mid \operatorname{ord}_{\mathbb{Z}_d}(d/d_i) = d_i \mid d$. To see $q_i > 1$, suppose to the contrary that some $q_j = 1$. Then $\prod_{i \neq j} q_i = d = \operatorname{lcm}(\{d_i\}_{i \neq j})$. But this contradicts the minimality of the index set I (and N) which we established w.l.o.g. in Claim A.4.3.

To see $N \leq \mathsf{primlmin}(D+1)$, observe that each q_i contributes different prime factors, and therefore $\mathsf{priml}(k) = \prod_{i=1}^{k} p_i \leq \prod_{i=1}^{k} q_i = d$, where p_i denotes the i-th prime number. Hence, if $\mathsf{priml}(k) \geq D+1$, then $d > D$, and therefore $k \leq \mathsf{primlmin}(D+1)$. □

Claim A.4.4 explains why $\beta = \min(|I|, \mathsf{primlmin}(D+1))$ is used in Lemma 3.3.12, because in Lemma 3.3.12 no assumptions on "minimality" of I (and N) were made (in particular $N > \mathsf{primlmin}(D+1)$ is possible).

Now, we change the distribution which we consider from $\gamma_i \xleftarrow{\$} [0, D]$ to $\gamma_i' \xleftarrow{\$} [0, q_i\lceil (D+1)/q_i \rceil - 1]$. (Observe that $q_i\lceil (D+1)/q_i\rceil \geq D+1$ is the smallest multiple of q_i which is larger or equal to $D+1$, and $\gamma_i' \bmod q_i$ is uniformly distributed.)

To simplify notation, let $\widehat{D} := D+1$, i.e. $\widehat{D}$ is the cardinality of $[0, D]$. One quickly computes

$$\rho_{\mathsf{sup}}(\gamma_i/\gamma_i') = \frac{1}{\widehat{D}} \cdot \left(\frac{1}{q_i\lceil \widehat{D}/q_i \rceil}\right)^{-1} = \frac{q_i\lceil \widehat{D}/q_i\rceil}{\widehat{D}} \leq 1 + \frac{q_i - 1}{\widehat{D}}$$

where we use that

$$0 \leq q_i\lceil \widehat{D}/q_i\rceil - \widehat{D} = (\widehat{D} \bmod q_i) \leq q_i - 1.$$

Observe that we can sample and write $\gamma_i' \xleftarrow{\$} [0, q_i\lceil (D+1)/q_i\rceil - 1]$ as

$$\gamma' = u_i + q_i v_i \quad \text{where} \quad u_i \xleftarrow{\$} [0, q_i - 1], \quad v_i \xleftarrow{\$} [0, \lceil \widehat{D}/q_i \rceil - 1].$$

For future reference, we record the following definitions and facts.

Claim A.4.5. *Let $u_i \xleftarrow{\$} [0, q_i - 1]$, $v_i \xleftarrow{\$} [0, \lceil \widehat{D}/q_i\rceil - 1]$ with q_i as above. Define*

$$S_u := \sum_{i=1}^{N} u_i m_i \frac{d}{d_i} \quad \text{and} \quad S_v := \sum_{i=1}^{N} v_i m_i \frac{q_i}{d_i} \tag{A.4.3}$$

Then $S_u \bmod d$ is uniform in $\mathbb{Z}_d$.

Proof. The claim is immediate by Lemma A.4.2 (and definition of q_i, u_i). □

Now, let

$$\rho := \rho_{\mathsf{sup}}((\gamma_1, \dots, \gamma_N)/(\gamma_1', \dots, \gamma_N')) \leq \prod_{i=1}^{N} (1 + \frac{q_i - 1}{\widehat{D}}) \tag{A.4.4}$$

Claim A.4.6. *It holds that $\rho \leq 4$.*

Proof. As Claim A.4.6 follows from unrelated technical computations, we prove this separately in Lemma A.4.10, which only needs the following constraints, already observed in (the proof of) Claim A.4.4,

- $1 < q_i \leq D$ for all $i = 1, \ldots, N$.
- $\prod_{i=1}^{N} q_i < D^2$ and for all subset products over $I' \subsetneq \{1, \ldots, N\}$ it holds that $\prod_{i \in I'} q_i \leq D$.

□

Now that notation and setup are in place, we turn to the following central claim.

Claim A.4.7. *It holds that*

$$\Pr\left[S \in [0, K]_{\mathbb{Z}_p} + \mu\right] \leq \rho \cdot \Pr\left[\frac{1}{d}(S_u \mod d) + S_v \in_{\mathbb{Z}_p} [0, K']_{\mathbb{Z}_p} + \mu'\right]$$

where

$$K' = K + 2\beta M \quad \textit{and} \quad \mu' = \mu - \beta M, \tag{A.4.5}$$

for $\beta = \min(N, \mathsf{primlmin}(D+1))$.

Proof. From our definition of γ_i' and Eq. (A.4.4), we get

$$\begin{aligned}
&\Pr\left[S \in [0, K]_{\mathbb{Z}_p} + \mu\right] \\
&= \Pr\left[\sum_{i=1}^{N} \gamma_i \cdot \frac{m_i}{d_i} \in_{\mathbb{Z}_p} [0, K]_{\mathbb{Z}_p} + \mu\right] \\
&\leq \rho \cdot \Pr\left[\sum_{i=1}^{N} \gamma_i' \cdot \frac{m_i}{d_i} \in_{\mathbb{Z}_p} [0, K]_{\mathbb{Z}_p} + \mu\right] \\
&= \rho \cdot \Pr\left[\sum_{i=1}^{N} (u_i + q_i v_i) \cdot \frac{m_i}{d_i} \in_{\mathbb{Z}_p} [0, K]_{\mathbb{Z}_p} + \mu\right] \\
&= \rho \cdot \Pr\left[\frac{1}{d} S_u + S_v \in_{\mathbb{Z}_p} [0, K]_{\mathbb{Z}_p} + \mu\right]
\end{aligned} \tag{A.4.6}$$

where we first use the properties of $\rho_{\sup}(\cdot/\cdot)$ to replace γ_i by γ_i', then we use $\gamma_i' = u_i + q_i v_i$ and rewrite the sum. As usual (by Eq. (A.4.2)), we have

$$\begin{aligned}
\frac{1}{d} S_u &= \frac{1}{d}(S_u \mod d) + \left\lfloor \frac{S_u}{d} \right\rfloor \\
&= \frac{1}{d}\left(\sum_{i=1}^{N} u_i \cdot m_i \frac{d}{d_i} \mod d\right) + \left\lfloor \sum_{i=1}^{N} u_i \cdot m_i \frac{1}{d_i} \right\rfloor.
\end{aligned}$$

We first derive a bound for $\left\lfloor \frac{S_u}{d} \right\rfloor$. Note that

$$\left\lfloor \sum_{i=1}^{N} u_i \cdot m_i \frac{1}{d_i} \right\rfloor \leq \left\lfloor \sum_{i=1}^{N} m_i \frac{q_i - 1}{d_i} \right\rfloor \leq \left\lfloor M \cdot \sum_{i=1}^{N} \frac{q_i - 1}{d_i} \right\rfloor \leq \left\lceil \sum_{i=1}^{N} \frac{q_i - 1}{d_i} \right\rceil \cdot M$$

where $\alpha = \lceil \sum_{i=1}^{N} \frac{q_i - 1}{d_i} \rceil \leq N$, since $(q_i - 1)/d_i < 1$ by choice of q_i (namely, $q_i \mid d_i$). Analogously,

$$\left\lfloor \sum_{i=1}^{N} u_i \cdot m_i \frac{1}{d_i} \right\rfloor \geq \left\lfloor -M \cdot \sum_{i=1}^{N} \frac{q_i - 1}{d_i} \right\rfloor \geq -\left\lceil \sum_{i=1}^{N} \frac{q_i - 1}{d_i} \right\rceil \cdot M$$

hence $-\alpha M$ is a lower bound. Thus, we find

$$\left\lfloor \frac{S_u}{d} \right\rfloor = \left\lfloor \sum_{i=1}^{N} u_i \cdot m_i \frac{1}{d_i} \right\rfloor \in [-\alpha M, \alpha M]. \tag{A.4.7}$$

That is, the possible values of the sum lie in the interval $[-\alpha M, \alpha M] = [0, 2\alpha M] - \alpha M$. Note that, by Claim A.4.4 and our simplifying assumptions on N and I in Claim A.4.3, $\alpha \leq \min(N, \mathsf{primlmin}(D+1)) = \beta$ holds.

Now, we can continue Eq. (A.4.6) with

$$\begin{aligned} &\rho \cdot \Pr\left[\frac{1}{d} S_u + S_v \in_{\mathbb{Z}_p} [0, K]_{\mathbb{Z}_p} + \mu\right] \\ &= \rho \cdot \Pr\left[\frac{1}{d}(S_u \bmod d) + \left\lfloor \frac{S_u}{d} \right\rfloor + S_v \in_{\mathbb{Z}_p} [0, K]_{\mathbb{Z}_p} + \mu\right] \\ &\leq \rho \cdot \Pr\left[\frac{1}{d}(S_u \bmod d) + S_v \in_{\mathbb{Z}_p} [0, K']_{\mathbb{Z}_p} + \mu'\right] \end{aligned}$$

where we first used Eq. (A.4.2) as usual, and then Eq. (A.4.7), as well as the definition $K' = K + 2\beta M$ and $\mu' = \mu - \beta M$. This proves Claim A.4.7. □

We are now in a position to prove Lemma 3.3.12. We first show Eq. (3.3.12) of Lemma 3.3.12.

Claim A.4.8. *It holds that*

$$\Pr\big[S \in [0, K]_{\mathbb{Z}_p} + \mu\big] \leq \rho \cdot \begin{cases} \frac{1}{d} & \text{if } d(K'+1) < p \\ \frac{1}{d} + 2\frac{K'+1}{p} & \text{always} \end{cases}$$

From Claim A.4.8 the first claim of the core lemma, Eq. (3.3.12), follows using $\rho \leq 4$ from Claim A.4.6.

Proof. Using Claim A.4.7, it suffices to prove that

$$\varepsilon := \Pr\left[\frac{1}{d}(S_u \bmod d) + S_v \in_{\mathbb{Z}_p} [0, K']_{\mathbb{Z}_p} + \mu'\right] \leq \frac{1}{d} + 2\frac{K'+1}{p}$$

and $\varepsilon \leq 1/d$ if $d(K'+1) < p$.

Since by construction, u_i and v_i are stochastically independent, we find

$$\varepsilon \leq \sum_{t \in \mathbb{Z}_p} \Pr\left[\frac{1}{d}(S_u \bmod d) \in_{\mathbb{Z}_p} [0, K']_{\mathbb{Z}_p} + \mu' - t\right] \cdot \Pr\big[S_v \equiv_p t\big].$$

Now, recall that $S_u \bmod d$ is uniformly distributed in $\mathbb{Z}_d$ (cf. Claim A.4.5), indeed, this was the reason for switching from γ_i to γ_i'. Thus,

$$\Pr\left[\frac{1}{d} S_u \in_{\mathbb{Z}_p} [0, K']_{\mathbb{Z}_p} + z'\right] = \Pr\left[\frac{1}{d} \cdot U_{\mathbb{Z}_d} \in_{\mathbb{Z}_p} [0, K']_{\mathbb{Z}_p} + z'\right] \leq \frac{1}{d} + 2\frac{K'+1}{p}$$

where $z' = \mu' - t$ and the inequality follows from Lemma 3.3.8. Hence,

$$\varepsilon \leq \sum_{t \in \mathbb{Z}_p} \frac{1}{d} \cdot \Pr\big[S_v \equiv_p t\big] = \frac{1}{d} + 2\frac{K'+1}{p}$$

and this part of the claim follows. Moreover, by Eq. (3.3.6) of Lemma 3.3.8, if $\lceil (K'+1)/\lfloor p/d \rfloor \rceil \leq 1$, then $\varepsilon \leq 1/d$ (by the same argument). Since $\lfloor x \rfloor \leq x$, we can simplify to $\lceil (K'+1)/\lfloor p/d \rfloor \rceil \leq \lceil d(K'+1)/p \rceil \leq 1$, and thus, $d(K'+1)/p \leq 1$, as claimed. □

Finally, we turn to proving Eq. (3.3.13) of Lemma 3.3.12.

Claim A.4.9. *Suppose $d \leq D$ and there is some i^* such that $dm_{i^*}/d_{i^*} > K' = K + 2\beta M$. Then*

$$\Pr\big[S \in_{\mathbb{Z}_p} [0,K]_{\mathbb{Z}_p} + \mu\big] \leq \frac{8}{D+1}.$$

Proof. Again, we continue from Claim A.4.7 and use independence of the u_is and v_is to find

$$\begin{aligned}
&\Pr\left[\frac{1}{d}(S_u \mod d) + S_v \in_{\mathbb{Z}_p} [0,K']_{\mathbb{Z}_p} + \mu'\right] \\
&= \sum_{t \in \mathbb{S}_d} \Pr\left[\frac{1}{d}(S_u \mod d) = t\right] \cdot \Pr\big[S_v \in_{\mathbb{Z}_p} [0,K']_{\mathbb{Z}_p} + \mu' - t\big] \\
&= \frac{1}{d} \sum_{t \in \mathbb{S}_d} \Pr\big[S_v \in_{\mathbb{Z}_p} [0,K']_{\mathbb{Z}_p} + \mu' - t\big]
\end{aligned}$$

where $\mathbb{S}_d = \{i/d \mod p \mid i \in [0, d-1]\} \subseteq \mathbb{Z}_p$. In the last equality, we use again that $S_u \mod d$ is uniform in $\mathbb{Z}_d$ by Claim A.4.5. Using the independence of the v_i's we further condition on all but i^*, and find

$$\begin{aligned}
&\frac{1}{d} \sum_{t \in \mathbb{S}_d} \Pr\big[S_v \in [0,K']_{\mathbb{Z}_p} + \mu' - t\big] \\
&= \frac{1}{d} \sum_{t \in \mathbb{S}_d} \Pr\left[\sum_{i=1}^{N} v_i \cdot m_i \frac{q_i}{d_i} \in [0,K']_{\mathbb{Z}_p} + \mu' - t\right] \\
&= \sum_{y \in \mathbb{Z}_p} \left(\Pr\left[\sum_{i \neq i^*} v_i \cdot m_i \frac{q_i}{d_i} = y\right] \cdot \right. \\
&\qquad \left. \underbrace{\frac{1}{d} \sum_{t \in \mathbb{S}_d} \Pr\left[v_{i^*} \cdot m_{i^*} \frac{q_{i^*}}{d_{i^*}} \in [0,K'] + \mu' - t - y\right]}_{(\star)} \right)
\end{aligned} \tag{A.4.8}$$

To improve readability, we abbreviate terms with index i^* as v^*, m^*, etc. Now, we want to apply Lemma 3.3.9 to $(\star)$, where $a \mathrel{\widehat{=}} q^* \cdot |m^*|$ and $b \mathrel{\widehat{=}} d^*$, and $D \mathrel{\widehat{=}} \lceil (D+1)/q^* \rceil - 1$, and $K \mathrel{\widehat{=}} K'$ and $\mu \mathrel{\widehat{=}} \mu' - y$. First, observe that the requirement

$$(K+1) + D\frac{a}{b} < \frac{1}{b} \cdot \left\lfloor \frac{p}{d/b} \right\rfloor$$

of Lemma 3.3.9 is satisfied when the corresponding variables are inserted (by our premise on K, N, D, p). Namely, since $\frac{1}{b} \cdot \lfloor \frac{p}{d/b} \rfloor \leq \frac{p}{d} - \frac{1}{b}$ and $\frac{1}{b} \leq 1$, it suffices to see $2 + K + D\frac{a}{b} \leq \frac{p}{d}$. With $d \leq D$, $\frac{a}{b} \leq M$ we arrive at $D(K' + DM + 2) < p$, which holds by assumption. Thus, by the conclusion of Lemma 3.3.9, we find

$$\begin{aligned}
&\frac{1}{d} \cdot \sum_{t \in \mathbb{S}_d} \Pr\left[v^* \cdot m^* \frac{q^*}{d^*} \in [0,K'] + \mu' - t - y\right] \\
&\leq \frac{1}{d} \cdot \left\lceil \frac{d^*(K'+1)}{m^* q^*} \right\rceil \frac{1}{\lceil (D+1)/q^* \rceil}.
\end{aligned}$$

Since, by assumption $dm^*/d^* < K'$, one can check that[2]

$$\frac{1}{d}\left\lceil\frac{d^*(K'+1)}{m^*q^*}\right\rceil\frac{1}{\lceil(D+1)/q^*\rceil} \leq \frac{\frac{q^*}{d}+\frac{d^*(K'+1)}{dm^*}}{D+1} \leq \frac{2}{D+1}.$$

Plugging that bound back into Eq. (A.4.8), we obtain

$$\frac{1}{d}\sum_{t\in\mathbb{S}_d}\Pr\left[S_v \in [0,K']_{\mathbb{Z}_p}+\mu'-t\right] \leq \frac{2}{D+1}$$

and the claim follows since

$$\Pr\left[S \in [\mu,\mu+K]_{\mathbb{Z}_p}\right] \leq \rho\cdot\frac{1}{d}\sum_{t\in\mathbb{S}_d}\Pr[S_v = t] \leq \frac{8}{D+1}.$$

□

This finishes the proof of Claim A.4.9, and hence of the Lemma 3.3.12. □

Lemma A.4.10. *Let $D, N \in \mathbb{N}$ and $q_i \in \mathbb{N}$ with $2 \leq q_i \leq D$ for $i = 1,\ldots,N$. Suppose that $\prod_{i=1}^N q_i < D^2$, that $q_1 \geq \ldots \geq q_N$, and that any subset product is at most D. Then*

$$\prod_{i=1}^N(1+\frac{q_i-1}{D+1}) \leq \prod_{i=1}^N(1+\frac{q_i}{D}) \leq 4$$

Unfortunately, we cannot provide much intuition for Lemma A.4.10 besides a proof overview: Namely, either $N = 2$, in which case the claim holds since $1 + q_i/D \leq 2$. Or $N > 2$. In that case, $\prod_{i=2}^N q_i \leq D$ and $q_2 \leq \sqrt{D}$ (since $q_1 \geq q_2$), by the "subset product premise". From this, it is easy to show that $\prod_{i=2}^N(1+\frac{q_i}{D}) \leq 2$ holds for "big enough" D (namely $D > 4$). The remaining cases (namely $D \leq 4$) are checked exhaustively.

Proof. We start with a simpler claim.

Claim A.4.11. *Let $a \geq b \geq 1$ and $\tau = ab$. Let $a' \geq b' \geq 1$ with $a' > a$ and $a'b' = \tau$. Then*

$$(1+a/D)(1+b/D) \leq (1+a'/D)(1+b'/D). \tag{A.4.9}$$

Proof. This follows by multiplying out both sides, subtracting the common term $1+\tau/D^2$ on both sides, and multiplying with D to obtain the equivalent condition $a+b \leq a'+b'$. Using $ab = a'b' = \tau$, this becomes

$$a+\tau/a \leq a'+\tau/a'$$

and for $f(x) = x+\tau/x$ it is readily seen that f is monotonely increasing on domain $[\sqrt{\tau},\infty)$. Thus, the claim follows from $a' > a$ and $a \geq \sqrt{\tau}$ (which holds since $ab = \tau$ and $a \geq b$). □

[2] Let $A = \frac{1}{d}\lceil\frac{d^*(K'+1)}{m^*q^*}\rceil$. Let $B = 1/\lceil(D+1)/q^*\rceil$. We have to show $A/B \leq 2/(D+1)$. First, note that $B \leq q^*/(D+1)$ since $1/\lceil x\rceil \leq 1/x$. Using $\lceil x\rceil \leq x+1$ in A, we find $A/B \leq \frac{1}{D+1}(\frac{d^*(K'+1)}{dm^*}+\frac{q^*}{d})$. Using $q^* \mid d^* \mid d$ and $d^* \neq d$ (since $N > 1$), we know $\frac{q^*}{d} \leq \frac{d^*}{d} \leq \frac{1}{2}$. Moreover, since $dm^*/d^* < K'$, i.e. $dm^*/d^* \leq K'+1$, holds by assumption, we find $dm^* \leq d^*(K'+1)$ and hence $\frac{d^*(K'+1)}{dm^*} \leq 1$. All in all, $A/B \leq (1+\frac{1}{2})/(D+1) \leq 2/(D+1)$.

The claim extends to products $\prod_{i=1}^{N}(1+\frac{a_i}{D})$ in the following manner: Consider $(a_1,\dots,a_N)$ with $a_1 \geq \dots \geq a_N \geq 1$ and $\prod_{i=1}^{N} a_i = \tau$ and $(a'_1,\dots,a'_N)$ with analogous constraints. Suppose that $(a_i)_i$ and $(a'_i)_i$ differ only in components j_1 and j_2 with $j_1 < j_2$, and that $a_{j_1} < a'_{j_1}$. Then $\prod_{i=1}^{N}(1+a_i/D) < \prod_{i=1}^{N}(1+\frac{a'_i}{D})$ by Claim A.4.11.

As a simple consequence, to maximize a product of the form $\prod_{i=1}^{N}(1+a_i/D)$ with constraints $a_i \in [2,M]_{\mathbb{R}}$ and $\prod_{i=1}^{N} a_i = \tau$, one must use a (permutation of) $(M,\dots,\tau/(2^{N-\ell-1}M^{\ell}),2,\dots,2)$, where ℓ is maximal.

Now, we return to prove the lemma. Let $\tau = \prod_{i=1}^{N} q_i$ and note that the product $\prod_{i=1}^{N}(1+\frac{q_i}{D})$ is maximized by maximizing q_1 and q_2 (due to $q_i \leq D$ and $q_1 q_2 \leq \tau < D^2$).

It is useful to deal with following special case first:

Claim A.4.12. *Suppose that $D \geq 5$, $q_1 \leq \sqrt{D}$ and $\prod_{i=1}^{N} q_i \leq D$. Then $\prod_{i=1}^{N}(1+\frac{q_i}{D}) \leq 2$.*

Proof. The claim evidently holds for $N = 1$. It also holds for $N = 2$. Indeed, for $N = 2$ and any (fixed) $q_1 \geq q_2$, setting $D = q_1 q_2$ is the worst case. For this, one obtains $(1+\frac{q_1}{D})(1+\frac{q_2}{D}) = \frac{q_1+1}{q_1}\frac{q_2+1}{q_2}$, and for $D \geq 5$, this is at most 2.[3] The claim also holds for arbitrary N if $5 \leq D \leq 15$.[4] Thus, suppose $D \geq 16$ and $N \geq 3$. By the discussion after Claim A.4.11, we find

$$\begin{aligned}\prod_{i=1}^{N}(1+\frac{q_i}{D}) &\leq \left(1+\frac{\sqrt{D}}{D}\right)\cdot\left(1+\frac{\sqrt{D}/2^{N-2}}{D}\right)\cdot\left(1+\frac{2}{D}\right)^{N-2}\\ &\leq \left(1+\frac{3/2\sqrt{D}+1}{D}\right)\cdot\left(1+\frac{2}{D}\right)^{N-2}\end{aligned}$$

where we maximized q_1 and q_2 (over $\mathbb{R}$) under the constraints that $q_i \geq 2$ and $\prod_{i=1}^{N} q_i \leq D$ and $q_1 \leq \sqrt{D}$ for all i. From $(1+x/k)^k \leq e^x$ for $x \geq 0$, $k \in \mathbb{N}$, we find

$$\left(1+\frac{2}{D}\right)^{N-2} = \left(\left(1+\frac{2}{D}\right)^{D}\right)^{(N-2)/D} \leq e^{2(N-2)/D}.$$

From, $D \geq \prod_{i=1}^{N} q_i \geq 2^N$, we get $N \leq \log(D)$, and thus $2(N-2)/D \leq 2(\log(D)-2)/D$. Moreover, $f(x) = 2(\log(x)-2)/x$ is maximized at $x = 4e \leq 11$, with $f(4e) \lessapprox 0.2654$ we find

$$e^{2(N-2)/D} \leq e^{2(\log(D)-2)/D} \leq e^{0.2654} \leq 4/3.$$

Furthermore $(1+\frac{3/2\sqrt{D}+1}{D})$ is monotonely decreasing, hence $(1+\frac{3/2\sqrt{D}+1}{D}) \leq 1+\frac{7}{16}$ for $D \geq 16$. Thus, for $D \geq 16$, we find

$$\prod_{i=1}^{N}(1+\frac{q_i}{D}) \leq \left(1+\frac{7}{16}\right)\cdot 4/3 < 2$$

This proves Claim A.4.12. □

[3] The claim does not hold for $D = 4$, as $(1+2/4)^2 = 9/4$. Since $D = 5$ is prime, only $q_1 = 5$, $q_2 = 1$ is possible, but this violates $q_2 \geq 2$, so there is nothing to check. For $D = 6$, choosing $q_1 = 3$, $q_2 = 2$ yields $(1+\frac{q_1}{D})(1+\frac{q_2}{D}) = 2$. Moreover, $\frac{q_1+1}{q_1}\frac{q_2+1}{q_2}$ only decreases for larger q_1 or q_2, so the claim holds for $q_1 q_2 > 6$ (with $q_1, q_2 \geq 2$) as well.

[4] Cases with 2 terms were already covered. Cases with more terms, i.e., $N = 3$, still work and are most easily checked programmatically. Cases with 4 or more terms are irrelevant since $2^4 = 16$ already exceeds 15.

Now, we prove the lemma by case distinction. First, note that it is easily verified for $D \leq 4$, so we can make use of Claim A.4.12 in the following. Since $(1 + q_1/D) \leq 2$ for any q_1, we only need to show that $\rho' = \prod_{i=2}^{N}(1 + q_i/D) \leq 2$. Moreover, we know, by the premise on subset products, that $\prod_{i=2}^{N} q_i \leq D$.

- Case: $q_2 \leq \sqrt{D}$. Then Claim A.4.12 applies to $q_2, \ldots, q_N$ and yields $\rho' \leq 2$.
- Case: $q_2 > \sqrt{D}$. Then $q_1 q_2 > D$, so $N = 2$, and $\rho' \leq (1 + q_2/D) \leq 2$.

This completes the proof. □

A.4.6. Proof of Theorem 3.3.3

Theorem 3.3.3. *Let* RAST *be the random affine shortness test with uniform distribution* $\mathcal{D}$ *over* $[0, D]^N$*, dimension* N*, range bound* K*, and any offset* $\mu \in \mathbb{Z}_p$*. Let* $K' = (1 + 2\beta)K$ *where* $\beta = \min(N, \mathsf{primlmin}(D + 1))$ *and suppose that* $2D(K' + DK + 2) < p$*. Then* RAST *is fractionally* (K', D)*-sound with error* $8/(D + 1)$*,*

Proof. Let $\vec{x} \in \mathbb{Z}_p^N$ and $\mu \in \mathbb{Z}_p$. We have to show that if $\vec{x}$ is not uniformly (K', D)-short, i.e. if there is no $d \in [1, D]$ with $d\vec{x} \in [-K', K']_{\mathbb{Z}_p}^N$, then $\Pr\Big[\mu + \sum_{i=1}^{N} x_i \gamma_i \in [0, K]_{\mathbb{Z}_p}\Big] \leq 8/(D + 1)$. Since this must hold for all μ, in particular $-\mu$, it is equivalent to show

$$\Pr\Big[\sum_{i=1}^{N} x_i \gamma_i \in [0, K]_{\mathbb{Z}_p} + \mu\Big] \leq 8/(D + 1).$$

We derive this inequality from the core lemma (Lemma 3.3.12). But in order to apply Lemma 3.3.12, we need that all x_i are of the form $x_i \equiv_p \frac{m_i}{d_i}$ with $|m_i| \leq M$ for suitable M. We choose $M = K$, so we have to show $x_i \in \mathbb{Q}_{K,D}$. Thus, we make a case distinction, based on following observation: Consider any fixed choice of $x_1, \ldots x_N \in \mathbb{Z}_p$. Suppose there are two distinct challenges $(\gamma_1, \ldots, \gamma_N)$, $(\gamma'_1, \ldots, \gamma'_N)$ which are accepting and differ only in the i-th component, i.e. $\gamma_j = \gamma'_j$ except for $j = i^*$, and w.l.o.g. $\gamma_{i^*} > \gamma'_{i^*}$. Let $\zeta \equiv_p \sum_{i=1}^{N} x_i \gamma_i$ and $\zeta' \equiv_p \sum_{i=1}^{N} x_i \gamma'_i$. Then $\zeta - \zeta' \equiv_p \sum_{i=1} x_i(\gamma_i - \gamma'_i) \equiv_p x_i(\gamma_{i^*} - \gamma'_{i^*})$. Thus $x_{i^*} \equiv_p \frac{\zeta - \zeta'}{\gamma_{i^*} - \gamma'_{i^*}} \in \mathbb{Q}_{K,D}$, since $\zeta - \zeta' \in [-K, K]$ and $\gamma_{i^*} - \gamma'_{i^*} \in [0, D]$. Now, we distinguish two cases.

Case 1: For every i^* there exist two accepting $(\gamma_j)_j \neq (\gamma'_j)_j$ which differ only in the i^*-th component. In that case, we argued above that $x_i \in \mathbb{Q}_{K,D}$ for every i. Thus, Lemma 3.3.12 is applicable with $M \mathrel{\widehat{=}} K$ and D. Moreover, we use that $D(K' + DM + 2) = D(K' + DK + 2) < p$, which is a premise of Theorem 3.3.3 (and also implies $D \cdot (K + 2\beta M) = D \cdot (1 + 2\beta)K < p$). The claim then follows from Lemma 3.3.12. (By choice of the index set I in Lemma 3.3.12, either the common denominator d of the x_is satisfies $D < d < D^2$, in which case Eq. (3.3.12) and $D^2(K' + 1) < p$ implies an error of at most $4/(D + 1)$, or $d \leq D$, in which case Eq. (3.3.13) implies an error of at most $8/(D + 1)$.)

Case 2: The opposite of Case 1, i.e. there exists some i^* for which no two accepting $(\gamma_j)_j \neq (\gamma'_j)_j$ which differ only in the i^*-th component exist. Then $\Pr[\sum_{i\neq i^*} c_i x_i + \gamma_{i^*} x_{i^*} \in [0,K] + \mu] \leq 1/(D+1)$ for any choice $c_i \in [0,D]$ for $i \neq i^*$. Thus, we get

$$\begin{aligned}
&\Pr\Big[\sum_{i\neq i^*} \gamma_i x_i + \gamma_{i^*} x_{i^*} \in [0,K] + \mu\Big] \\
= &\sum_{c_i \in [0,D], i \neq i^*} \Pr[\forall i \neq i^* \colon \gamma_i = c_i] \cdot \Pr\Big[\sum_{i\neq i^*} c_i x_i + \gamma_{i^*} x_{i^*} \in [0,K] + \mu\Big] \\
\leq &\sum_{c_i \in [0,D], i \neq i^*} \left(\frac{1}{D+1}\right)^{N-1} \cdot \frac{1}{D+1} \\
= &\frac{1}{D+1}
\end{aligned}$$

Thus, the probability ε that the test (falsely) accepts satisfies $\varepsilon \leq 1/(D+1)$. The claim follows. □

A.5. Security Reductions

A.5.1. Security Proof of $\mathsf{Sharp}_{\mathsf{GS}}$

In the following theorem, we show that $\mathsf{Sharp}_{\mathsf{GS}}$ is secure.

Theorem A.5.1. *The scheme* $\mathsf{Sharp}_{\mathsf{GS}}$ *has correctness error at most* $1 - [(1-\mathsf{p_r})^3 \cdot (1-\mathsf{p_x})^{4N}]^R$. *It is non-abort SHVZK under the SEI assumption. Suppose now that* $2(B\Gamma^2+1)L < p$ *and* $18K^2 < q$ *with* $K = (B\Gamma+1)L$. *Then it has relaxed soundness under the DLOG and SEI assumptions in* $\mathbb{G}_{\mathsf{com}}$ *and* $\mathbb{G}_{\mathsf{3sq}}$ *with knowledge error* $(\frac{2}{\Gamma+1})^R$ *for the relation*

$$\begin{aligned}
\mathcal{R}_{\mathsf{Ext}} = \Big\{(x_1,\dots,x_N,r) \colon & C_x = r_x G_0 + \sum_{i=1}^{N} x_i G_i \\
& \wedge \exists m_i, d \in \mathbb{Z} \colon x_i \equiv_q \frac{m_i}{d} \wedge -\frac{1}{4B} \leq \frac{m_i}{d} \leq B + \frac{1}{4B} \\
& \wedge |m_i| \leq (B\Gamma+1)L_x \wedge 1 \leq d \leq \Gamma\Big\}.
\end{aligned}$$

Concretely, with the hash-optimization, we have following reductions:

- *For every adversary* $\mathcal{A}$ *against non-abort SHVZK, there are adversaries* $\mathcal{B}_{\mathbb{G}_{\mathsf{com}}}, \mathcal{B}_{\mathbb{G}_{\mathsf{3sq}}}$ *whose runtime is roughly that of* $\mathcal{A}$ *and so that* $\mathsf{Adv}^{\text{na-hvzk}}_{\mathcal{A}} \leq \mathsf{Adv}^{\mathsf{sei}}_{\mathbb{G}_{\mathsf{com}},\mathcal{B}_{\mathbb{G}_{\mathsf{com}}}} + R \cdot \mathsf{Adv}^{\mathsf{sei}}_{\mathbb{G}_{\mathsf{3sq}},\mathcal{B}_{\mathbb{G}_{\mathsf{3sq}}}}$.
- *For every adversary* $\mathcal{A}$ *against knowledge which runs at most* T *steps, there are adversaries* $\mathcal{B}_{CR}$, $\mathcal{B}_{\mathbb{G}_{\mathsf{com}}}, \mathcal{B}_{\mathbb{G}_{\mathsf{3sq}}}$ *whose* expected *runtime is bounded roughly by* $3 \cdot R \cdot T$, *and so that* $\mathsf{Adv}^{\mathsf{ke}}_{\mathcal{A}} \leq (\frac{2}{\Gamma+1})^R + \mathsf{Adv}^{\mathsf{crhf}}_{\mathsf{Hash},\mathcal{B}_{CR}} + \mathsf{Adv}^{\mathsf{dlog}}_{\mathbb{G}_{\mathsf{com}},\mathcal{B}_{\mathbb{G}_{\mathsf{com}}}} + \mathsf{Adv}^{\mathsf{dlog}}_{\mathbb{G}_{\mathsf{3sq}},\mathcal{B}_{\mathbb{G}_{\mathsf{3sq}}}}$.
- *For witness relation* $\mathcal{R}_{\mathsf{Ext}} \vee \mathcal{R}^{\mathbb{G}_{\mathsf{com}}}_{\mathsf{Bind}} \vee \mathcal{R}^{\mathbb{G}_{\mathsf{3sq}}}_{\mathsf{Bind}} \vee \mathcal{R}_{\mathsf{Coll}}$, *where* $\mathcal{R}^{G}_{\mathsf{Bind}}$ *is a binding-break relation in group* G *(i.e. a non-trivial DLOG relation), and* $\mathcal{R}_{\mathsf{Coll}}$ *is a non-trivial collision for* Hash, *the knowledge error is* $(\frac{2}{\Gamma+1})^R$.

To be precise, we consider the S*-bounded SEI assumption in* $\mathbb{G}_{\mathsf{com}}$ *and* $\mathbb{G}_{\mathsf{3sq}}$.

Proof. Throughout this proof, we have $i \in [1,N]$, $j \in [1,3]$, $k \in [1,R]$.

Correctness. As $x_i, y_{i,j} \in [0, B]$ and $\gamma_k \in [1, \Gamma]$, we have $z_{k,i}, z_{k,i,j} \in [0, (B\Gamma + 1)L_x]$ (see Section 3.2.4), unless masking aborts. The other checks of the verifier succeed due to the homomorphic properties of MPed and the fact that $z_{k,i} = \gamma_k \cdot \widetilde{x}_{k,i} + x_{k,i}$, etc., holds if the prover did not abort. It is then easy to see, that all of the verifier's checks pass.

Honest-Verifier Zero-Knowledge. We define a simulator for valid transcripts as follows. On input of the public parameters and the statement (C_x, B), the simulator Sim proceeds as follows:

- Sample $\gamma_k \xleftarrow{\$} [0, \Gamma]$
- Set $C_y := r_y G_0$ for $r_y \xleftarrow{\$} [0, S]$
- Set $C_{k,*} := r^*_k H_0$ for $r^*_k \xleftarrow{\$} [0, S]$
- Set $z_{k,i} = \mathsf{mask}_x(0, \widetilde{x}_{k,i})$ and $z_{k,i,j} = \mathsf{mask}_x(0, \widetilde{y}_{k,i,j})$ for $\widetilde{x}_{k,i}, \widetilde{y}_{k,i,j} \xleftarrow{\$} \mathsf{R}_x$
- Set $t_{k,x} = \mathsf{mask}_r(0, \widetilde{r}_{k,x})$, $t_{k,y} = \mathsf{mask}_r(0, \widetilde{r}_{k,y})$ and $t^*_k = \mathsf{mask}_r(0, \widetilde{r}^*_k)$ for $\widetilde{r}_{k,x}, \widetilde{r}_{k,y}, \widetilde{r}^*_k \xleftarrow{\$} \mathsf{R}_r$
- If any masking failed, then abort, i.e. output $\perp$.
- Compute $D_{k,x} = -\gamma_k C_x + t_{k,x} G_0 + \sum_{i=1}^{N} z_{k,i} G_i$ and $D_{k,y} = -\gamma_k C_y + t_{k,y} G_0 + \sum_{i=1}^{N} \sum_{j=1}^{3} z_{k,i,j} G_{i,j}$
- Compute $f^*_{k,i} = 4 z_{k,i}(\gamma_k B - z_{k,i}) + \gamma_k^2 - \sum_{j=1}^{3} z_{k,i,j}^2$
- Compute $D_{k,*} = -\gamma_k C_{*,k} + t^*_k H_0 + \sum_{i=1}^{N} f^*_{k,i} H_i$
- Set $\Delta = \mathsf{Hash}(\{D_{k,x}, D_{k,y}, D_{k,*}\}_{k=1}^{R})$
- Output $\Delta, C_y, C_{k,*}, \gamma_k, z_{k,i}, z_{k,i,j}, t_{k,x}, t_{k,y}, t_{k,*}$

It is easy to check that the output of Sim is indistinguishable from non-aborting real transcripts. We do so in game hops.

$\mathsf{Game}\ 1$: Output a transcript tr from an interaction of an honest verifier and prover from the definition of $\mathsf{Sharp}_{\mathsf{GS}}$. If the transcript is aborting, output $\perp$ instead.

$\mathsf{Game}\ 2$: Act as in *game 1* but instead of computing $D_{k,x}, D_{k,y}$ and $D_{k,*}$ as in the real protocol, compute them as Sim. A quick computation shows that *game 1* and *game 2* are perfectly indistinguishable.

$\mathsf{Game}\ 3$: Act as in *game 2* but instead of sampling $z_{k,i}, z_{k,i,j}, t_{k,x}, t_{k,y}, t_{k,*}$ as in the real protocol, sample them as Sim, i.e. via $\mathsf{mask}(0, \cdot)$. The games *game 2* and *game 3* are statistically indistinguishable. Namely, their statistical distance is bounded by $RN(1 + 3)\varepsilon_x + R\varepsilon_r$, where the masking errors ε_x and ε_r correspond to the masking schemes mask_x and mask_r (see Section 3.2.4). Due to uniform rejection sampling, $\varepsilon_x = \varepsilon_r = 0$. (Note that, if, say $z_{k,i} = \perp$ we cannot define the corresponding $D_{k,x}$. This is not a problem, since we consider non-abort SHVZK, hence a transcript where $z_{k,i} = \perp$ is replaced by $\perp$, both in game 2 and 3.)

$\mathsf{Game}\ 4$: Instead of computing the commitments $C_y, C_{k,*}$ as in the real protocol, compute them as Sim. *Game 3* and *game 4* are indistinguishable under the hiding property of the commitment scheme. More precisely, the we need 1 reduction to the SEI assumption in $\mathbb{G}_{\mathsf{com}}$ for C_y, and R in $\mathbb{G}_{\mathsf{3sq}}$ for $C_{k,*}$.

As the output of *game 4* is equal to the output of Sim, $\mathsf{Sharp}_{\mathsf{GS}}$ is non-abort SHVZK.

Soundness. We assume that we are given a number of accepting related transcripts, and first show the statement for a single repetition, i.e. $R = 1$. After that, we discuss how repetitions change the security proofs and how to obtain the transcripts. For readability, we omit k (denoting the current repetition) in the following as index from the transcripts. Assume that a PPT adversary can interactively produce three valid transcripts tr, tr', tr'' with fixed first message Δ, C_y, C_*, and distinct challenges γ, γ', γ'' and masked terms $[z_i, z_{i,j}, t_x, t_y, t_*]$, $[z'_i, z'_{i,j}, t'_x, t'_y, t'_*]$ and $[z''_i, z''_{i,j}, t''_x, t''_y, t''_*]$. We define F_x, F'_x, F''_x as in the verification, similarly for F_y, F_*, f^*_i. We denote by $\overline{X}$ and $\underline{X}$ the differences $X' - X$ and $X'' - X$ respectively for $X \in [\gamma_k, z_i, z_{i,j}, t_x, t_y, t_*, F_x, F_y, F_*, f_i]$. Without loss of generality, $\overline{\gamma}, \underline{\gamma} > 0$. Note that $p = \mathrm{ord}(\mathbb{G}_{\mathsf{com}})$ and $q = \mathrm{ord}(\mathbb{G}_{3\mathsf{sq}})$.

Step 1 – Opening the Commitments: First, we extract openings of C_x, C_y, C_*. By collision resistance of Hash, we have $D_x := F_x = F'_x = F''_x$. Further, the check of the verifier guarantees:

$$\begin{aligned} D_x &= -\gamma C_x + t_x G_0 + \sum_{i \in [1,N]} z_i G_i \\ &= -\gamma' C_x + t'_x G_0 + \sum_{i \in [1,N]} z'_i G_i \end{aligned}$$

Rearranging this equation leads to the following equality:

$$\begin{aligned} \overline{\gamma} C_x &= \overline{t_x} G_0 + \sum_{i \in [1,N]} \overline{z_i} G_i \\ \implies C_x &= \overline{t_x}/\overline{\gamma} G_0 + \sum_{i \in [1,N]} \overline{z_i}/\overline{\gamma} G_i. \end{aligned}$$

Thus, C_x commits to values $x_i \equiv_p \overline{z_i}/\overline{\gamma} \in \mathbb{Z}_p$. With the same calculation, we can show that $x_i \equiv_p \underline{z_i}/\underline{\gamma}$. Note that x_i is well defined as MPed is binding. (We reduce to DLOG in $\mathbb{G}_{\mathsf{com}}$ and $\mathbb{G}_{3\mathsf{sq}}$ for binding.[5]) Similarly, we find openings for the remaining commitments

$$\begin{aligned} C_y &= \overline{t_y}/\overline{\gamma} G_0 + \sum_{i \in [1,N], j \in [1,3]} \overline{z_{i,j}}/\overline{\gamma} G_{i,j} \text{ and} \\ C_* &= \overline{t^*}/\overline{\gamma} H_0 + \sum_{i \in [1,N]} \overline{f^*_i}/\overline{\gamma} H_i. \end{aligned}$$

So C_y commits to values $y_{i,j} \equiv_p \overline{z_{i,j}}/\overline{\gamma} \equiv_p \underline{z_{i,j}}/\underline{\gamma} \in \mathbb{Z}_p$ and C_* to $\alpha^*_{1,i} \equiv_q \overline{f^*_i}/\overline{\gamma} \equiv_q \underline{f^*_i}/\underline{\gamma} \in \mathbb{Z}_q$.

Step 2 – Decomposition: We now show that the three-square decomposition holds and that $[x_i]_{\mathbb{Q}}$ is indeed in the range $[-\frac{1}{4B}, B + \frac{1}{4B}]_{\mathbb{Q}}$. We proceed similarly to [CKLR21b] but our proof is more subtle, as we argue over two different groups with incompatible (prime) modulus. Nonetheless, we can conclude $[x_i]_{\mathbb{Q}} \in [-\frac{1}{4B}, B + \frac{1}{4B}]_{\mathbb{Q}}$ since the rational representative is unique in both groups.

First, we define $\widehat{x_i} \equiv_q \overline{z_i}/\overline{\gamma} \equiv_q \underline{z_i}/\underline{\gamma}$. Note that $\widehat{x_i}$ is computed modulo q, but since all values are short we find

$$\begin{aligned} \overline{z_i}/\overline{\gamma} \equiv_p \underline{z_i}/\underline{\gamma} &\implies \overline{z_i}\underline{\gamma} \equiv_p \underline{z_i}\overline{\gamma} \implies \overline{z_i}\underline{\gamma} = \underline{z_i}\overline{\gamma} \text{ over } \mathbb{Z} \\ &\implies \overline{z_i}\underline{\gamma} \equiv_q \underline{z_i}\overline{\gamma} \implies \overline{z_i}/\overline{\gamma} \equiv_q \underline{z_i}/\underline{\gamma}. \end{aligned}$$

In particular, $\widehat{x_i}$ is well-defined (modulo q) and as short rationals, we have $x_i = \widehat{x_i}$.

[5] Note that due to random self-reducibility of DLOG, we need not incur a loss of N in the reduction.

Now, we set $\widetilde{x_i} \equiv_q z_i - \gamma \widehat{x_i}$. Using the definition of $\widehat{x_i}$, we have

$$\begin{aligned} \widetilde{x_i} &\equiv_q z_i - \gamma\widehat{x_i} \equiv_q z_i + z_i' - z_i' - \gamma\widehat{x_i} \equiv_q -\overline{z_i} + z_i' - \gamma\widehat{x_i} \\ &\equiv_q \overline{\gamma}\widehat{x_i} + z_i' - \gamma\widehat{x_i} \equiv_q (\gamma' - \gamma)\widehat{x_i} + z_i' - \gamma\widehat{x_i} \equiv_q z_i' - \gamma'\widehat{x_i}. \end{aligned}$$

Also, $\widetilde{x_i} \equiv_q z_i'' - \gamma''\widehat{x_i}$ follows accordingly. We similarly set $\widehat{y_{i,j}} \equiv_q \overline{z_{i,j}}/\overline{\gamma}$ and $m_{i,j} \equiv_q z_{i,j} - \gamma\widehat{y_{i,j}} \equiv_q z_{i,j}' - \gamma'\widehat{y_{i,j}} = z_{i,j}'' - \gamma''\widehat{y_{i,j}}$, where the equalities follow as above. Inserting these equalities and interpreting f_i^* (similarly $(f_i^*)'$, $(f_i^*)''$) as a polynomial with variable γ, we obtain:

$$\begin{aligned} f_i^* &\equiv_q \gamma^2[4\widehat{x_i}(B - \widehat{x_i}) + 1 - \sum_{j\in[1,3]} \widehat{y_{i,j}}^2] + \gamma\alpha_{1,i} + \alpha_{0,i}, \\ (f_i^*)' &\equiv_q (\gamma')^2[4\widehat{x_i}(B - \widehat{x_i}) + 1 - \sum_{j\in[1,3]} \widehat{y_{i,j}}^2] + \gamma'\alpha_{1,i} + \alpha_{0,i} \\ (f_i^*)'' &\equiv_q (\gamma'')^2[4\widehat{x_i}(B - \widehat{x_i}) + 1 - \sum_{j\in[1,3]} \widehat{y_{i,j}}^2] + \gamma''\alpha_{1,i} + \alpha_{0,i} \end{aligned}$$

for some appropriate $\alpha_{1,i}, \alpha_{0,i}$. (In fact, $\alpha_{1,i}^* = \alpha_{1,i}$.) We can subtract the first from the second (third) equation and then divide the resulting equation by $\overline{\gamma}$ ($\underline{\gamma}$) respectively. This leads to:

$$\begin{aligned} \overline{f_i^*}/\overline{\gamma} &\equiv_q (\gamma' + \gamma)[4\widehat{x_i}(B - \widehat{x_i}) + 1 - \sum_{j\in[1,3]} \widehat{y_{i,j}}^2] + \alpha_{1,i}, \\ \underline{f_i^*}/\underline{\gamma} &\equiv_q (\gamma'' + \gamma)[4\widehat{x_i}(B - \widehat{x_i}) + 1 - \sum_{j\in[1,3]} \widehat{y_{i,j}}^2] + \alpha_{1,i}. \end{aligned}$$

As $\alpha_{1,i}^* \equiv_q \overline{f_i^*}/\overline{\gamma} \equiv_q \underline{f_i^*}/\underline{\gamma}$ (see first step), we obtain:

$$\begin{aligned} &(\gamma'' - \gamma')[4\widehat{x_i}(B - \widehat{x_i}) + 1 - \sum_{j\in[1,3]} \widehat{y_{i,j}}^2] \equiv_q 0 \\ \implies &4\widehat{x_i}(B - \widehat{x_i}) + 1 \equiv_q \sum_{j\in[1,3]} \widehat{y_{i,j}}^2 \\ \implies &4\overline{z_i}(\overline{\gamma}B - \overline{z_i}) + \overline{\gamma}^2 \equiv_q \sum_{j\in[1,3]} \overline{z_{i,j}}^2 \end{aligned}$$

Setting $K = (B\Gamma + 1)L_x$ and noting $|\overline{z_i}| \leq K$, we find $|4\overline{z_i}(\overline{\gamma}B - \overline{z_i}) + \overline{\gamma}^2| \leq 4K^2 + 4K^2 + K^2 < q/2$ and $\sum_{j\in[1,3]} \overline{z_{i,j}}^2 \leq 3K^2 < q/2$. Thus, the equation holds over the integers and as a result, it holds that $4\overline{z_i}(\overline{\gamma}B - \overline{z_i}) + \overline{\gamma}^2 \geq 0$. Dividing by $\overline{\gamma}$ yields $4\frac{\overline{z_i}}{\overline{\gamma}}i(B - \frac{\overline{z_i}}{\overline{\gamma}}) + 1 \geq 0$. Now, Lemma A.2.2 implies that $\frac{\overline{z_i}}{\overline{\gamma}} \in [-\frac{1}{4B}, B + \frac{1}{4B}]_{\mathbb{Q}}$.

Step 3 – Repetitions: Consider a setting with repetitions. Suppose we are given three related transcripts such that in repetition k, the challenges $\gamma_k, \gamma_k', \gamma_k''$ are pairwise distinct. Then the previous steps apply, and we conclude the same soundness guarantees. Note that it suffices to have such related transcripts for *any* of the k repetitions. Further, since the *same* commitment C_x is used in all iterations, extractions $x_{k,i}$ (of x_i) for differing k must coincide, or the binding property and hence DLOG is broken in $\mathbb{G}_{\mathrm{com}}$.

Step 4 – Obtaing the transcripts: It is well-known how to obtain related transcripts, but we give a brief sketch for the sake of completeness. First, run the (malicious) prover with a random challenge. If the honest verifer rejects, the extractor has nothing to do; it just outputs this view. So assume otherwise. If $R = 1$, rewind the (malicious) prover and try (fresh) random challenges (without repetition) until 3 transcripts are found or all challenges exhausted. For $R > 1$, a very naive strategy exploits that the protocol is $(2^R + 1)$-special sound, but this degrades the knowledge error. A less wasteful approach works with about $3R$ expected rewinds. The basic idea is to not pick all challenges γ_k fresh, but keep all but one fixated, and do that for all $k = 1, \dots, R$ in parallel. See [BBC+18; ACK21] for concrete examples.

□

A.5.2. Proof of $\mathsf{Sharp}_{\mathsf{SO}}^{\mathsf{Po}}$

In this section, we prove the security of $\mathsf{Sharp}_{\mathsf{SO}}^{\mathsf{Po}}$. As usual, we consider the optimized variant which uses a CRHF.

Theorem A.5.2. *The scheme* $\mathsf{Sharp}_{\mathsf{SO}}^{\mathsf{Po}}$ *has correctness error at most* $1 - (1 - 1/L)^{3+2R+4N}$. *It is non-abort SHVZK under the SEI assumption. Let* $K' = (1+2\beta)K$ *where* $K = (B\Gamma+1)L$ *and* $\beta = \min(4N, \mathsf{primlmin}(\Gamma+1))$. *Suppose* $9(K')^2 < q/2$ *and* $(\Gamma + 1)^R - 1 < p$. *Then* $\mathsf{Sharp}_{\mathsf{SO}}^{\mathsf{Po}}$ *has relaxed soundness under the DLOG and SEI in* $\mathbb{G}_{\mathsf{com}}$ *and* $\mathbb{G}_{3\mathsf{sq}}$ *with knowledge error* $\frac{2+8^R}{(\Gamma+1)^R}$ *for relation*

$$\begin{aligned}\mathcal{R}_{\mathsf{Ext}} = \Big\{&(x_1, \dots, x_N, r) \colon C_x = r_x G_0 + \sum_{i=1}^{N} x_i G_i \\ &\wedge \exists m_i, d \in \mathbb{Z} \colon x_i \equiv_q \frac{m_i}{d} \wedge -\frac{1}{4B} \le \frac{m_i}{d} \le B + \frac{1}{4B} \\ &\wedge |m_i| \le K' \wedge 1 \le d \le \Gamma\Big\}.\end{aligned}$$

Concretely, with the hash-optimization, we have following reductions:

- *For every adversary* $\mathcal{A}$ *against non-abort SHVZK, there are adversaries* $\mathcal{B}_{\mathbb{G}_{\mathsf{com}}}$, $\mathcal{B}_{\mathbb{G}_{3\mathsf{sq}}}$ *whose runtime is roughly that of* $\mathcal{A}$ *and so that* $\mathsf{Adv}_{\mathcal{A}}^{\text{na-hvzk}} \le \mathsf{Adv}_{\mathbb{G}_{\mathsf{com}},\mathcal{B}_{\mathbb{G}_{\mathsf{com}}}}^{\mathsf{sei}} + R \cdot \mathsf{Adv}_{\mathbb{G}_{3\mathsf{sq}},\mathcal{B}_{\mathbb{G}_{3\mathsf{sq}}}}^{\mathsf{sei}}$.
- *For every adversary* $\mathcal{A}$ *against knowledge which runs at most* T *steps, there are adversaries* $\mathcal{B}_{CR}$, $\mathcal{B}_{\mathbb{G}_{\mathsf{com}}}$, $\mathcal{B}_{\mathbb{G}_{3\mathsf{sq}}}$ *whose* expected *runtime is bounded roughly by* $6 \cdot R \cdot T$, *and so that* $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{ke}} \le \frac{2+8^R}{(\Gamma+1)^R} + \mathsf{Adv}_{\mathsf{Hash},\mathcal{B}_{CR}}^{\mathsf{crhf}} + \mathsf{Adv}_{\mathbb{G}_{\mathsf{com}},\mathcal{B}_{\mathbb{G}_{\mathsf{com}}}}^{\mathsf{dlog}} + \mathsf{Adv}_{\mathbb{G}_{3\mathsf{sq}},\mathcal{B}_{\mathbb{G}_{3\mathsf{sq}}}}^{\mathsf{dlog}}$.
- *For witness relation* $\mathcal{R}_{\mathsf{Ext}} \vee \mathcal{R}_{\mathsf{Bind}}^{\mathbb{G}_{\mathsf{com}}} \vee \mathcal{R}_{\mathsf{Bind}}^{\mathbb{G}_{3\mathsf{sq}}} \vee \mathcal{R}_{\mathsf{Coll}}$, *where* $\mathcal{R}_{\text{DL-rel}}^{G}$ *is a non-trivial DLOG relation in group* G, *and* $\mathcal{R}_{\mathsf{Coll}}$ *is a non-trivial collision for* Hash, *the knowledge error is* $\frac{2+8^R}{(\Gamma+1)^R}$.

The rest of this section consists of a proof of Theorem A.5.2.

Correctness. Correctness follows by a straightforward check. Whenever an honest prover does not abort (due to masking), the honest verifier will accept.

Non-Abort SHVZK This follows almost as for $\mathsf{Sharp}_{\mathsf{GS}}$ in Theorem A.5.1. Namely, Phase 2 can be argued identically (except, that there are no repetitions now). Once Phase 2 is replaced by a simulation, Phase 1 can be simulated by using $x_i = 0$ instead of the real witness. Since ζ_k are masked terms, this incurs k masking errors, which however are 0 for uniform rejection sampling.

Soundness. The rest of this section is dedicated to proving the soundness error.

A.5.2.1. Step 1: Extracting Phase 2.

We begin as described in the outline. Let G_0 be the knowledge soundness game from Definition 2.3.16. As the first step, define G_1 which only differs from G_0 by replacing the malicious prover with an extractor in Phase 2. More concretely, note that Phase 2 is a 3-special sound Σ-protocol for the relation (where $y_{i,0} \coloneqq x_i$)

$$\begin{aligned}\mathcal{R}_{\mathrm{Ext}} = \big\{(&C_x, C_y, \{\zeta\}_{k\in[1,R]}; \{x_i\}_{i\in[1,N]}, \{y_{i,j}\}_{i\in[1,N],j\in[0,3]},\\ &r_x, r_y, \{\mu_k\}_{k\in[1,R]}) :\\ &C_x = r_x G_0 + \sum_{i=1}^{N} x_i G_i\\ &\wedge \forall k \in [1,R]\colon \ \sum_{i=1}^{N}\sum_{j=0}^{3} y_{i,j}\gamma_{i,j} + \mu_k = \zeta_k\\ &\wedge C_y = r_y G_0 + \sum_{i=1}^{N}\sum_{j=1}^{3} y_{i,j}G_{i,j} + \sum_{k=1}^{R}\mu_k \widetilde{G}_k\\ &\wedge \forall i \in [1,N]\colon \ 1 + 4x_i(B - x_i) \equiv_q \sum_{j\in[1,3]} y_{i,j}^2\big\}.\end{aligned}$$

or a hash-collision or DLOG relation, i.e. $\mathcal{R}_{\mathrm{Ext}} \vee \mathcal{R}_{\mathrm{Bind}}^{\mathbb{G}_{\mathrm{com}}} \vee \mathcal{R}_{\mathrm{Bind}}^{\mathbb{G}_{\mathrm{3sq}}} \vee \mathcal{R}_{\mathrm{Coll}}$. This follows analogously to Theorem A.5.1 for $\mathrm{Sharp}_{\mathrm{GS}}$, up to standard changes. Thus, as in Theorem A.5.1, we find that, the runtime changes from strict runtime t_0 to *expected* time $t_1 \approx 3t_0$ and the knowledge error is $2/(\Gamma+1)^R$. In game G_1, we return 1 iff the extraction succeeded as well. Overall, it follows that

$$\Pr[\mathrm{G}_0 = 1] \leq \Pr[\mathrm{G}_1 = 1] + 2/(\Gamma+1)^R \quad \text{and} \quad t_1 \approx 3t_0.$$

A.5.2.2. Step 2: Extracting Phase 1.

Recall that Phase 2 of the protocol is actually a proof of knowledge for $\mathcal{R}_{\mathrm{Ext}}$ with statement $(C_x, C_y, \{\zeta_k\}_k)$. In game G_1, we always try to extract Phase 2, so now, we are almost in the setting of (random affine) shortness testing. The main difference is, that in the latter setting, the choice of $(\{x_i\}, \{y_{i,j}\})$ would be *fixed beforehand*, whereas in our case, it is only *committed to*. Thus, we need to account for the case of a binding break.

Looking ahead, the completed extractor works as follows:

1. Pick a uniform challenge $\gamma_{i,j}^{(k)} \xleftarrow{\$} [0,\Gamma]$ ($i \in [1,N]$, $j \in [0,3]$, $k \in [1,R]$) for Phase 1 and run the extractor for Phase 2.
2. If extraction (of Phase 2) fails, also output failure, i.e. $\perp_{\mathrm{ext}}$.
3. If the verifier did not accept the run, output the generated view. (There is nothing to do.)
4. If the extracted witness $(y_{i,j})_{i,j}$ (where $y_{i,0} = x_i$) is of the form $y_{i,j} = \frac{m_{i,j}}{d}$ for $d \in [1,\Gamma]$ and $m_i \in [0,K']$ with $K' = (1+2\beta K)$, output $(x_i)_i$.[6] In this case, $x_i = \frac{m_i}{d} \in [0,B]$ as claimed.
5. Else, the extracted $(x_i)_i$ are "invalid". Try to obtain a DLOG relation as follows:

[6] Note that we need an efficient algorithm to check this. But as noted in Remark 3.2.5, for any choice of $M, D \in \mathbb{N}$ with $MD < p/2$ we can efficiently [FSW03] compute (m,d) for $x \equiv_p \frac{m}{d}$ if $x \in \mathbb{Q}_{M,D}$. In our setting, $M = K'$ and $D = \Gamma$ satisfies $\Gamma K' < q/2$ by assumption.

- Rewind before sending the challenge (in Phase 1) and pick fresh uniform challenges $\widetilde{\gamma}_{i,j}^{(k)} \xleftarrow{\$} [0,\Gamma]$ (with repetition) for Phase 1 and run the extractor for Phase 2. Repeat until again a witness $(y'_{i,j})_{i,j}$ is output.
- If $(y_{i,j})_{i,j} \neq (y'_{i,j})_{i,j}$, return the non-trivial DLOG relation corresponding to $(y_{i,j} - y'_{i,j})_{i,j}$.
- Otherwise, output failure, i.e. $\perp_{\text{ext}}$.

Let us now analyze this extraction and the soundness of the protocol. Extraction failure (item 2) and verifier rejection (item 3) are trivial to account for. So let us assume that a witness $(\{x_i\},\{y_{i,j}\})$ was extracted from Phase 2. Each test $\sum_{i=1}^{N}\sum_{j=0}^{3}\gamma_i^{(k)} y_{i,j} + \mu_k \in [0,K]$ (where $y_{i,0} = x_i$) is a random affine shortness test (Definition 3.3.2), where μ_k plays the role of the constant offset μ and $4N$ elements are checked at once. The parameters of this test are dimension $N \mathrel{\widehat{=}} 4N$, and range bound K, test distribution $U_{[0,\Gamma]^{4N}}$ and offset μ_k. As shown in Theorem 3.3.3, the test is fractional (K',Γ)-sound with error $\delta_{\text{snd}} \leq 8/(\Gamma+1)$, where $K' = (1+2\beta)K$ and $\beta = \min(4N, \mathsf{primlmin}(\Gamma+1))$. The probability to cheat (in all of them) is therefore $\delta_{\text{snd}}^R \leq (\frac{8}{\Gamma+1})^R$. Now there are two cases:

Case 1 (Item 4): There exists some $d \in [1,\Gamma]$ such that $dy_{i,j} \in [-K',K']$ for all $i \in [1,N]$, $j \in [0,3]$, where $y_{i,0} := x_i$. For this case, consider the quadratic relations in Phase 2, which are known to hold over $\mathbb{Z}_q$ (for all i):

$$1 + 4x_i(B - x_i) \equiv_q \sum_{j\in[1,3]} y_{i,j}^2$$
$$\iff d^2 + 4dx_i(dB - dx_i) \equiv_q \sum_{j\in[1,3]} (dy_{i,j})^2$$

Since and $dy_{i,j} \in [-K',K']$ (where $y_{i,0} = x_i$), the left-hand side has absolute value at most $\Gamma^2 + 4K'(B\Gamma + K') \leq 9(K')^2 < q/2$. The right-hand-side is at most $3(K')^2 < q/2$. Thus, the right equality holds over the integers, and the left equality holds over the rationals. Consequently, we find $x_i \in [-\frac{1}{4}B, B + \frac{1}{4}B]_{\mathbb{Q}}$ (by Lemma A.2.2). Since additionally $dx_i \in [-K',K']$ for all i, we have found a witness for $\mathcal{R}_{\text{Ext}}$, which completes this case.

Case 2 (Item 5): There is no $d \in [1,\Gamma]$ such that $dy_{i,j} \in [-K',K']$ for all i,j, but the shortness tests failed to catch this. In this case, the extractor rewinds and retries Phase 1 (with fresh challenges and still running the extractor for Phase 2) until a second extracted run is found; denote the extracted witness by $(y'_{i,j})_{i,j}$. Let $\varepsilon_1 = \Pr[\mathrm{G}_1 = 1]$. It is easy to check that the expected number of retries is 1 and that, overall, the expected time t_2 for the extractor is roughly bounded by $2t_1 \leq 6t_0$. Since the soundness error of the repeated shortness test is δ_{snd}^R, with probability at least $(\varepsilon_1 - \delta_{\text{snd}}^R)/\varepsilon_1$, it happens that $(y_{i,j})_{i,j} \neq (y'_{i,j})_{i,j}$ or $\mu_k \neq \mu'_k$ for the second accepting transcipt. In that case, a non-trivial DLOG relation can be derived from the binding break, i.e. the two different witnesses.[7]

Define G_2 as a run of the complete extractor, and let it output 1 if and only if the verifier was convinced in the initial run and a valid witness is outputted. Note that G_0 resp. G_2 now correspond to the real resp. ideal executions in the definition of knowledge soundness (Definition 2.3.16). We see that, except with probability at most $\varepsilon_1 \cdot \frac{\delta_{\text{snd}}^R}{\varepsilon_1} = \delta_{\text{snd}}^R$, where $\varepsilon_1 = \Pr[\mathrm{G}_1 = 1]$, game G_2 succeeds in producing a valid witness $\mathcal{R}_{\text{Ext}} \vee \mathcal{R}_{\text{Bind}}^{\mathbb{G}_{\text{com}}} \vee \mathcal{R}_{\text{Bind}}^{\mathbb{G}_{3\text{sq}}} \vee \mathcal{R}_{\text{Coll}}$. Thus, overall we find

$$\Pr[\mathrm{G}_2 = 1] \leq \Pr[\mathrm{G}_1 = 1] + \delta_{\text{snd}}^R \leq \Pr[\mathrm{G}_0 = 1] + 2/(\Gamma+1)^R + \delta_{\text{snd}}^R$$

[7] The complete extracted witnesses $\vec{w}, \vec{w}'$ also contain components r_x, r_y.

If the extractor does not fail, it returns a witness for $\mathcal{R}_{\mathsf{Ext}} \vee \mathcal{R}^{\mathbb{G}_{\mathrm{com}}}_{\mathrm{Bind}} \vee \mathcal{R}^{\mathbb{G}_{3\mathrm{sq}}}_{\mathrm{Bind}} \vee \mathcal{R}_{\mathrm{Coll}}$, where $\mathcal{R}^{G}_{\mathrm{Bind}} = \mathcal{R}^{G}_{\mathrm{DL\text{-}rel}}$ is a binding break, i.e. a non-trivial DLOG relation in $G \in \{\mathbb{G}_{\mathrm{com}}, \mathbb{G}_{3\mathrm{sq}}\}$, and $\mathcal{R}_{\mathrm{Coll}}$ is a non-trivial collision for Hash. The knowledge error for this witness relation is $2/(\Gamma+1)^R + \delta^R_{\mathrm{snd}} \leq (2+8^R)/(\Gamma+1)^R$, as claimed in last item of Theorem A.5.2. Witnesses for $\mathcal{R}^{\mathbb{G}_{\mathrm{com}}}_{\mathrm{DLog}}, \mathcal{R}^{\mathbb{G}_{3\mathrm{sq}}}_{\mathrm{DLog}}$ and $\mathcal{R}_{\mathrm{Coll}}$ can instead be viewed as adversaries against DLOG and collision resistance, showing the second item. This completes the proof of knowledge soundness.

A.5.3. Security Proof of $\mathsf{Sharp}_{\mathsf{HO}}$

Here, we prove the security of Theorem A.3.1.

Theorem A.3.1. *Let* Sample *be a sampling algorithm for* $\mathbb{G}$*. The scheme* $\mathsf{Sharp}^{+\mathsf{HO}}_{\mathsf{GS}}$ *has correctness error at most* $1-(1-\mathsf{p_r}')^N[(1-\mathsf{p_r})^3 \cdot (1-\mathsf{p_x})^N]^R$*. It is non-abort SHVZK under the SEI assumptions on* $\mathbb{G}$ *and the SEI and the SI assumptions on* $\mathbb{H}$*. Is has relaxed soundness for the relation*

$$\begin{aligned}\mathcal{R}_{\mathsf{Ext}} = \Big\{(x_1,\ldots,x_N,r)\colon & C_x = r_x G_0 + \sum_{i=1}^{N} x_i G_i \\ & \wedge \exists m_i \in \mathbb{Z}, k \in \mathbb{N}_0\colon -\frac{1}{4B} \leq \frac{m_i}{e^k} \leq B + \frac{1}{4B} \\ & \wedge x_i \equiv_q \frac{m_i}{e^k} \wedge |m_i| \leq (B\Gamma+1)L_x \wedge 1 \leq e^k \leq \Gamma\Big\}.\end{aligned}$$

under the DLOG, SEI assumptions on $\mathbb{G}$*, and the DLOG, SEI, SI, assumption and hardness of* (Γ, e)*-relaxed DLOG-relations in* $\mathbb{H}$*, where all asumptions are all w.r.t. to* Sample*. The knowledge error is* $(\frac{2}{\Gamma+1})^R$*. Concretely, with the hash-optimization, we have following reductions:*

- *For every adversary* $\mathcal{A}$ *against non-abort SHVZK, there are adversaries* $\mathcal{B}_{\mathbb{G},SEI}, \mathcal{B}_{\mathbb{H},SEI}, \mathcal{B}_{\mathbb{H},SI}$ *whose runtime is roughly that of* $\mathcal{A}$ *and so that* $\mathsf{Adv}^{\text{na-hvzk}}_{\mathcal{A}} \leq \mathsf{Adv}^{\text{sei}}_{\mathbb{H},\mathcal{B}_{\mathbb{G}_{\mathrm{com}},SEI}} + \mathsf{Adv}^{\text{sei}}_{\mathbb{H},\mathcal{B}_{\mathbb{H},SEI}} + \mathsf{Adv}^{\text{si}}_{\mathbb{H},\mathcal{B}_{\mathbb{H},SI}}$.
- *For every adversary* $\mathcal{A}$ *against knowledge which runs at most* T *steps, there are adversaries* $\mathcal{B}_{CR}$*,* $\mathcal{B}_{\mathbb{G},DLOG}, \mathcal{B}_{\mathbb{H},DLOG}, \mathcal{B}_{\mathbb{H},SI}, \mathcal{B}_{\mathbb{H},rlxDLOG}$*, whose* expected *runtime is roughly* $3 \cdot R \cdot T$*, and so that* $\mathsf{Adv}^{\mathrm{ke}}_{\mathcal{A}} \leq (\frac{2}{\Gamma+1})^R + \mathsf{Adv}^{\text{crhf}}_{\mathsf{Hash},\mathcal{B}_{CR}} + \mathsf{Adv}^{\text{dlog}}_{\mathbb{G},\mathcal{B}_{\mathbb{G},DLOG}} + \mathsf{Adv}^{\text{dlog}}_{\mathbb{H},\mathcal{B}_{\mathbb{H},DLOG}} + \mathsf{Adv}^{\text{rlx-dlog}}_{\mathbb{H},(\Gamma,e,N),\mathcal{B}_{\mathbb{H},rlxDLOG}}$.

To be precise, we consider the S*-bounded SEI assumption in* $\mathbb{G}$ *and the* S*-bounded SEI assumption in* $\mathbb{H}$*.*

The analogous adaption of Theorem A.5.1 holds for $\mathsf{Sharp}^{+\mathsf{HO}}_{\mathsf{PoSO}}$*, where the same additional terms for reductions in* $\mathbb{H}$ *appear.*

Proof. Here, we demonstrate correctness, soundness and zero-knowledge of $\mathsf{Sharp}^{+\mathsf{HO}}_{\mathsf{GS}}$ in more detail. We note that all assumptions are w.r.t. to Sample, in particular, we assume the adversary has access to the random coins ρ_i used to generate the hidden order group elements in the CRS. (This is not the case for elements of $\mathbb{G}_{\mathrm{com}}$ and $\mathbb{G}_{3\mathrm{sq}}$. There, we still assume invertible sampling.)

Correctness. The rejection probability is increased by a factor of $(1 - \mathsf{p_r}') \leq (1 - 1/L)$ due to the additional masking of t_x'. It is straightforward to see that all "old" checks will pass, as the computations and checks for z_i, F_i are unmodified. The modified computation of the hash of Δ will pass, if $D_x' = F_x'$ holds. Hence, it remains to show that $D_x' = F_x'$, i.e.

$$\begin{aligned} D_x' &= \widetilde{r}_x' G_0' + \sum_{i=1}^{N} (\Gamma + 1)^{k-1} \widetilde{x}_i' G_i' \\ &\overset{!}{=} -\gamma' C_x' + t_x' \cdot G_0' + \sum_{i \in [1,N]} z_i' \cdot G_i' \\ &= F_x' \end{aligned}$$

holds, where $\gamma' = \sum_{k=1}^{R} \gamma_k (\Gamma + 1)^{k-1} \in [0, (\Gamma + 1)^R - 1]$, and $\widetilde{x}_i' = \sum_{k=1}^{R} (\Gamma + 1)^{k-1} \widetilde{x}_{k,i}$ and $z_i' = \sum_{k=1}^{R} (\Gamma + 1)^{k-1} \cdot z_{k,i}$. We have

$$\begin{aligned} F_x' &= -\gamma' C_x' + t_x' \cdot G_0' + \sum_{i \in [1,N]} z_i' \cdot G_i' \\ &= -\gamma' C_x' + (\gamma' r_x' + \widetilde{r}_x') \cdot G_0' + \sum_{i \in [1,N]} z_i' \cdot G_i' \\ &= \widetilde{r}_x' \cdot G_0' + (-\gamma' C_x' + \gamma' r_x' + \sum_{i \in [1,N]} z_i' \cdot G_i') \end{aligned}$$

Plugging in $\gamma' = \sum_{k=1}^{R} (\Gamma + 1)^{k-1} \cdot \gamma_k$, we find that

$$\begin{aligned} &- \gamma' C_x' + \gamma' r_x' + \sum_{i \in [1,N]} z_i' \cdot G_i' \\ &= \sum_{k=1}^{R} (\Gamma + 1)^{k-1} \left(-\gamma_k C_x' + \gamma_k r_x' \cdot G_0' + \sum_{i \in [1,N]} z_{k,i} \cdot G_i' \right) \\ &= \sum_{k=1}^{R} (\Gamma + 1)^{k-1} \sum_{i=1}^{N} (-\gamma_k x_i + z_{k,i}) G_i' \\ &= \sum_{k=1}^{R} (\Gamma + 1)^{k-1} \sum_{i=1}^{N} \widetilde{x}_{k,i} G_i' \end{aligned}$$

since by construction $C_x' = r_x' G_0 + \sum_{i=1}^{N} x_i G_i'$ and $z_{k,i} = \gamma_k x_i + \widetilde{x}_{k,i}$. Thus,

$$F_x' = \widetilde{r}_x' G_0' + \sum_{k=1}^{R} \sum_{i=1}^{N} (\Gamma + 1)^{k-1} \widetilde{x}_{k,i} G_i' = D_x'.$$

Soundness. The argument for soundness of $\mathsf{Sharp}_{\mathsf{GS}}^{\mathsf{+HO}}$ is basically the same as for $\mathsf{Sharp}_{\mathsf{GS}}$ in Theorem A.5.1, except, that the properties of the MPed commitment in $\mathbb{H}$ must be exploited additionally.

Let $\widehat{\Gamma} = (\Gamma + 1)^k - 1$. Observe that, by construction, the synthetic challenge γ' is uniform in $[0, \widehat{\Gamma}]$. Moreover, the synthesized proof of short opening is almost the same of the *simple PoSO*,[8] and our soundness argument as well, with the only difference being the choice of masking. Namely, the

[8] That is, usual the Σ-protocol for opening with short challenge and shortness check, as used in $\mathsf{Sharp}_{\mathsf{GS}}$ for example.

distributions of the mask $\widetilde{x}'_i$ is not the usual one. However, for soundness, the distribution of the mask does not matter at all (it may be adversarially chosen anyway). Consequently, the argument for the PoSO in Theorem A.5.1 applies without change. That is, either two[9] accepting transcripts tr and $\widehat{tr}$ with same first message but different challenges yield witnesses $x'_i = \frac{z'_i - \widehat{z}'_i}{\gamma' - \widehat{\gamma}'}$ of the form $x'_i = a_i/2^{e_i}$ for $e_i \geq 0$, $a_i \in \mathbb{Z}$ or a (Γ, e)-relaxed DLOG relation in $\mathbb{H}$ was found. By assumption, finding a (Γ, e, N)-relaxed DLOG relation w.r.t. Sample is hard. (Note we use a hat $\widehat{\cdot}$ to distinguish the transcripts, since primes $\cdot'$ are already used to indicate elements of our augmentation.)

Recall that we argued in particular, that in each iteration,

- either $\gamma_k = \widehat{\gamma}_k$ and $z_{k,i} = \widehat{z}_{k,i}$, i.e. this repetition "does not extract", or
- we extract $x_{k,i}$ and $x_i = x_{k,i}$ is unique for all "extracted" repetitions k.

or a non-trivial DLOG relation was found. Now, we have to show that $x'_i = x_i$, i.e. the extracted witness of the synthesized hidden order proof of small opening coincides with the other extractions. For this, note that

$$\begin{aligned} x'_i &= \frac{z'_i - \widehat{z'}_i}{\gamma' - \widehat{\gamma'}} = \frac{\sum_{k=1}^{R} (\Gamma + 1)^{k-1} (z_{k,i} - \widehat{z}_{k,i})}{\sum_{k=1}^{R} (\Gamma + 1)^{k-1} (\gamma_k - \widehat{\gamma}_k)} \\ &= \frac{\sum_{k=1}^{R} (\Gamma + 1)^{k-1} (x_{k,i}(\gamma_k - \widehat{\gamma}_k))}{\sum_{k=1}^{R} (\Gamma + 1)^{k-1} (\gamma_k - \widehat{\gamma}_k)} \\ &= x_i \cdot \frac{\sum_{k=1}^{R} (\Gamma + 1)^{k-1} (\gamma_k - \widehat{\gamma}_k)}{\sum_{k=1}^{R} (\Gamma + 1)^{k-1} (\gamma_k - \widehat{\gamma}_k)} = x_i \end{aligned}$$

where we used that $x_{k,i} = x_i$ for all k.[10] Thus, $x'_i = x_i$ and the extracted witnesses of all repetitions coincide. This finishes the proof.

For $\mathsf{Sharp}^{+\mathsf{HO}}_{\mathsf{PoSO}}$, an analogous reasoning applies, though simpler since "synthesized" variables are not needed.

Non-Abort SHVZK The simulator works as the simulator in Theorem A.5.1 (resp. Theorem A.5.2), with following additional steps:

1. Compute $\gamma' = \sum_{k=1}^{R} \gamma_i (\Gamma + 1)^{k-1}$.
2. Set $C'_x \xleftarrow{\$} \mathbb{H}$.
3. Let $z'_i = \sum_{k=1}^{R} (\Gamma + 1)^{k-1} \cdot z_{k,i}$ (using the simulated $z_{k,i}$).
4. Set $t'_x = \mathsf{mask}_{r'}(0, \widetilde{r}')$.
5. If masking fails, then abort, i.e. output $\perp$.
6. Compute $D'_x = -\gamma' C'_x + t'_x G'_0 + \sum_{i=1}^{N} z'_i G'_i$.
7. Adapt the output to include the additional messages.

[9] To show that x_i is of the form $x' = a/2^e$, two transcripts suffice. The soundness of the full argument still needs three transcripts.

[10] Strictly speaking, if $\gamma_k = \widehat{\gamma}_k$, then $x_{k,i}$ is not defined. By our assumption, we may assume it exists and equals x_i. This is a mere simplification, as the contribution of repetition k to the sum is 0 anyway, since (again by assumption) $z_{k,i} - \widehat{z}_{k,i} = 0$.

Table A.1.: Overview of parameters (where $S = 2^{256} - 1$ always) and proof sizes of variants of Sharp_{GS} in Bytes with correctness error $1 - p_{\text{succ}}$. We give the proof size with the 3-square decomposition (π), the amortized proof size (π_{amor}), the proof size with the 4-square decomposition (π_{4sqr}), the proof size for the augmentation with an additional RSA group element (π_{RSA}) and the proof size for the augmentation with an additional class group element (π_{CL}).

λ	κ	B	Γ	L	N	p	q	R	p_{succ}	π	π_{amor}	π_{4sqr}	π_{RSA}	π_{CL}
128	40	32	41	10	1	256	256	1	0.993	234	234	244	667	455
128	40	32	41	10	4	256	256	1	0.982	358	90	400	792	579
128	40	32	41	10	8	256	256	1	0.966	524	66	607	958	745
128	40	32	41	10	16	256	256	1	0.937	856	54	1022	1290	1077
128	40	64	41	10	1	256	256	1	0.993	250	250	264	683	471
128	40	64	41	10	4	256	256	1	0.982	422	106	480	856	643
128	40	64	41	10	8	256	256	1	0.966	652	82	767	1086	873
128	40	64	41	10	16	256	256	1	0.937	1112	70	1342	1546	1333
128	80	32	81	10	1	256	256	1	0.993	254	254	269	687	475
128	80	32	81	10	4	256	256	1	0.982	438	110	500	872	659
128	80	32	81	10	8	256	256	1	0.966	684	86	807	1118	905
128	80	32	81	10	16	256	256	1	0.937	1176	74	1422	1610	1397
128	80	64	81	10	1	256	315	1	0.993	285	285	304	718	505
128	80	64	81	10	4	256	315	1	0.982	517	130	595	950	738
128	80	64	81	10	8	256	315	1	0.966	827	104	982	1260	1048
128	80	64	81	10	16	256	315	1	0.937	1447	91	1757	1880	1668
128	128	32	129	10	1	301	347	1	0.993	318	318	339	751	538
128	128	32	129	10	4	301	347	1	0.982	574	144	660	1007	795
128	128	32	129	10	8	301	347	1	0.966	916	115	1087	1349	1137
128	128	32	129	10	16	301	347	1	0.937	1600	100	1942	2033	1821
128	128	64	129	10	1	333	411	1	0.993	360	360	385	793	580
128	128	64	129	10	4	333	411	1	0.982	664	166	766	1097	885
128	128	64	129	10	8	333	411	1	0.966	1070	134	1273	1503	1291
128	128	64	129	10	16	333	411	1	0.937	1882	118	2288	2315	2103

It is easy to check that the output is indistinguishable from non-aborting real transcripts. The justification is almost identical to the one in Theorem A.5.1. Namely, starting from the honest computation, first compute D'_x is in step 6 above (with otherwise honest values). This change is only conceptual. Then, compute t'_x as in step 4 above. Finally, an additional step is required to justify the switch from computing $C'_x = r'_x G'_0 + \sum_{i=1}^{N} x_i G'_i$ to $C'_x \xleftarrow{\$} \mathbb{H}$. Since r'_x is not used anymore, we can reduce this to SI and SEI assumptions (w.r.t. Sample). By SEI we can replace the term $A = r'_x G'_0$ by $A \xleftarrow{\$} \langle G'_0 \rangle$. Then, by SI we can replace $A \xleftarrow{\$} \langle G'_0 \rangle$ by $A \xleftarrow{\$} \mathbb{H}$. Now, C'_x is uniform distributed in $\mathbb{H}$. So we can sample $C'_x \xleftarrow{\$} \mathbb{H}$ instead. This is done by the simulator in step 2, and indeed, this game is the simulation, completing the proof. □

A.6. Additional Tables

Here, we provide some tables with an overview of the parameters and proof sizes of Sharp_{GS} in Table A.1 and $\text{Sharp}_{\text{SO}}^{\text{Po}}$ in Table A.2.

Table A.2.: Overview of parameters (where $S = 2^{256} - 1$ always) and proof sizes of variants of $\mathrm{Sharp}_{\mathrm{SO}}^{\mathrm{Po}}$ in Bytes with correctness error $1 - p_{\mathrm{succ}}$. We give the proof size with the 3-square decomposition (π), the amortized proof size (π_{amor}), the proof size with the 4-square decomposition (π_{4sqr}), the proof size for the augmentation with an additional RSA group element (π_{RSA}) and the proof size for the augmentation with an additional class group element (π_{CL}).

λ	κ	B	Γ	L	N	p	R	p_{succ}	π	π_{amor}	π_{RSA}	π_{CL}
128	40	32	43	10	1	256	1	0.991	300	300	734	521
128	40	32	43	10	4	256	1	0.980	556	139	989	777
128	40	32	43	10	8	256	1	0.964	896	112	1329	1117
128	40	32	43	10	16	256	1	0.935	1576	99	2010	1797
128	40	64	43	10	1	256	1	0.991	324	324	758	545
128	40	64	43	10	4	256	1	0.980	628	157	1061	849
128	40	64	43	10	8	256	1	0.964	1032	129	1465	1253
128	40	64	43	10	16	256	1	0.935	1840	115	2274	2061
128	80	32	43	10	1	256	2	0.989	323	323	757	544
128	80	32	43	10	4	256	2	0.978	579	145	1013	800
128	80	32	43	10	8	256	2	0.963	920	115	1353	1141
128	80	32	43	10	16	256	2	0.933	1600	100	2034	1821
128	80	64	43	10	1	256	2	0.989	355	355	789	576
128	80	64	43	10	4	256	2	0.978	659	165	1093	880
128	80	64	43	10	8	256	2	0.963	1064	133	1497	1285
128	80	64	43	10	16	256	2	0.933	1872	117	2306	2093
128	128	32	67	10	1	256	2	0.989	335	335	769	556
128	128	32	67	10	4	256	2	0.978	591	148	1025	812
128	128	32	67	10	8	256	2	0.963	932	117	1365	1153
128	128	32	67	10	16	256	2	0.933	1612	101	2046	1833
128	129	64	46	10	1	256	3	0.987	389	389	822	609
128	128	64	35	10	4	256	4	0.974	714	179	1148	935
128	128	64	35	10	8	256	4	0.959	1119	140	1553	1340
128	128	64	35	10	16	256	4	0.929	1928	121	2362	2149
128	40	32	1	10	1	256	40	0.919	777	777	–	–
128	40	32	1	10	16	256	40	0.866	2092	131	–	–
128	40	64	1	10	1	256	40	0.919	1113	1113	–	–
128	40	64	1	10	16	256	40	0.866	2668	167	–	–
128	128	32	1	10	1	256	128	0.773	1877	1877	–	–
128	128	32	1	10	16	256	128	0.729	3280	205	–	–
128	128	64	1	10	1	256	128	0.773	2917	2917	–	–
128	128	64	1	10	16	256	128	0.729	4560	285	–	–

B. Appendix for Chapter 4

B.1. From Batch Proofs to Folding

We briefly discuss how one can interpet the folding technique of [BCC+16; BBB+18] as a form of composing batch verification protocols. We only consider a commitment key $[\boldsymbol{g}] \in \mathbb{G}^{1\times m}$ and the statement $\rtimes \boldsymbol{w}\colon [\boldsymbol{g}]\boldsymbol{w} = [\boldsymbol{t}]$, and fix $k = 2$ as reduction factor for simplicity. The idea is as follows:

Step 1 (Preparation of Statement). Notice that the statement $[\boldsymbol{g}]\boldsymbol{w} = [\boldsymbol{t}]$ is not directly batch verifiable — it is a single statement. Thus, as a first step, split the statement $\rtimes \boldsymbol{w}\colon [\boldsymbol{g}]\boldsymbol{w} = [\boldsymbol{t}]$ into $[\boldsymbol{g}] = \left[\begin{smallmatrix} \boldsymbol{g}_0 \\ \boldsymbol{g}_1 \end{smallmatrix}\right]$, $\boldsymbol{w} = \left(\begin{smallmatrix} \boldsymbol{w}_0 \\ \boldsymbol{w}_1 \end{smallmatrix}\right)$ and let $[\boldsymbol{t}_i] = [\boldsymbol{g}_i]\boldsymbol{w}_i$. Observe that proving $\rtimes \boldsymbol{w}_i\colon [\boldsymbol{g}_i]\boldsymbol{w}_i = [\boldsymbol{t}_i]$ for $i \in \{0,1\}$ implies the original claim if $[\boldsymbol{t}_0] + [\boldsymbol{t}_1] = [\boldsymbol{t}]$. However, a batch proof for the two statements (for $i = 0$ and 1) is still not possible, as the matrices disagree ($[\boldsymbol{g}_0] \neq [\boldsymbol{g}_1]$). Fortunately, this can be salvaged by introducing redundancy, namely, by considering

$$\rtimes \boldsymbol{w}_j\colon \begin{bmatrix} \boldsymbol{g}_0 \\ \boldsymbol{g}_1 \end{bmatrix} \boldsymbol{w}_j = [\boldsymbol{v}_j]$$

where $[\boldsymbol{v}_0] = \left[\begin{smallmatrix} \boldsymbol{t}_0 \\ \boldsymbol{g}_1\boldsymbol{w}_0 \end{smallmatrix}\right]$ and $[\boldsymbol{v}_1] = \left[\begin{smallmatrix} \boldsymbol{g}_0\boldsymbol{w}_1 \\ \boldsymbol{t}_1 \end{smallmatrix}\right]$. In total, $\{[\boldsymbol{g}_i]\boldsymbol{w}_j\}_{i,j\in\{0,1\}}$ must be sent additionally to the verifier.[1] But now, both statements use the same matrix $[\boldsymbol{G}] = \left[\begin{smallmatrix} \boldsymbol{g}_0 \\ \boldsymbol{g}_1 \end{smallmatrix}\right]$.

Step 2 (Batch-Verification of Statements). By using a batch proof of knowledge, e.g. Protocol 4.3.12, the statement

$$\rtimes \boldsymbol{w}_i\colon [\boldsymbol{G}]\boldsymbol{w}_i = [\boldsymbol{v}_i] \tag{B.1.1}$$

is reduced to

$$\rtimes \widehat{\boldsymbol{w}}\colon [\boldsymbol{G}]\widehat{\boldsymbol{w}} = [\widehat{\boldsymbol{v}}] \tag{B.1.2}$$

where $\widehat{\boldsymbol{w}} \in \mathbb{F}_p^{m/2}$, i.e. the dimension of the witness was halved. However, as a trade-off, the height of the matrix was doubled, i.e. $[\boldsymbol{G}] \in \mathbb{G}^{2\times m/2}$ whereas $[\boldsymbol{g}] \in \mathbb{G}^{1\times m}$.

Step 3 (Batch-Verification Within a Statement). At this point, we batch together the rows of $[\boldsymbol{G}]$, e.g. with Protocol 4.3.8. As a result, the verification of

$$\rtimes \boldsymbol{w}\colon [\boldsymbol{G}]\widehat{\boldsymbol{w}} = [\widehat{\boldsymbol{v}}] \tag{B.1.3}$$

Eq. (B.1.2) is reduced to

$$\rtimes \boldsymbol{w}\colon [\widehat{\boldsymbol{G}}]\widehat{\boldsymbol{w}} = [\widehat{\boldsymbol{t}}]$$

where $[\boldsymbol{G}] \in \mathbb{G}^{2\times m/2}$ and $[\widehat{\boldsymbol{v}}] \in \mathbb{G}^2$, and $[\widehat{\boldsymbol{G}}] \in \mathbb{G}^{1\times m/2}$ and $[\widehat{\boldsymbol{t}}] \in \mathbb{G}$.

[1] Since $\sum_{i\in\{0,1\}} [\boldsymbol{g}_i]\boldsymbol{w}_i = [\boldsymbol{g}]\boldsymbol{w}$, it suffices to send 3 instead of 4 elements.

Recursion. Notice that after Step 3, the statement is similar to Step 1, namely $[\widehat{g}]\widehat{w} = [\widehat{t}]$ (with $[\widehat{g}] \mathrel{\widehat{=}} [\widehat{G}]$) except that the dimension of $\widehat{w}$ is half of $\widehat{w}$. As such, by recursively applying these steps, we obtain a logarithmic-size argument of knowledge of a preimage or a non-trivial kernel element of $[g]$.

Security Analysis. The security of this scheme is relatively straightforward. (It does not provide HVZK, only knowledge.) Namely, it follows by composing the security of the subprotocols, Protocol 4.3.12 and Protocol 4.3.8, and it is also clear that short-circuit extraction is inherited.

Comparison to Folding. The (optimized) folding approach can be viewed as specialization of the above protocol, where the challenges in Steps 2 and 3 are not independent.

B.2. An Efficient Proof of Correctness of a Shuffle

A proof of correctness of a shuffle is a proof that two (multi-)sets of ciphertexts decrypt to the same multi-set of plaintexts. This is especially interesting in settings with rerandomisable ciphertexts, because the "shuffling party" does not need to decrypt. For electronic voting, a shuffle achieves a certain unlinkability between the originally encrypted votes, and the (in a final step) decrypted votes, while the proofs of correctness of the shuffle ensure that the voting result is unaffected.

With our tools, it is possible to prove the correctness of a shuffle in logarithmic communication for ElGamal ciphertexts in a very straightforward manner. Namely, we commit to a permutation matrix (as part of $\boldsymbol{w}$) and rerandomisation randomness for the ElGamal ciphertexts (also part of $\boldsymbol{w}$). Then we prove that $[\boldsymbol{A}]\boldsymbol{w} = [\vec{\boldsymbol{c}}]$, where $[\boldsymbol{A}]$ is constructed from the old ciphertexts and the ElGamal public key, and $[\vec{\boldsymbol{c}}]$ is the vector of shuffled ciphertexts. We also add a proof that (the relevant part of) $\boldsymbol{w}$ commits to a permutation matrix, as sketched in Section 4.3.6. This all neatly fits into our framework, giving a logarithmic size proof overall. However, there is a huge drawback: The size of the permutation matrix, hence $\boldsymbol{w}$, is N^2 for N ciphertexts. Thus, the *computation* grows quadratically in N. This is unacceptable in practice.

Remark B.2.1 (Related work). Shuffle arguments already appear in [BBB+18]; they are built from a sorting circuit and comparing the sorted sequences. In an independent work [AVY] implementing Bulletproofs, an improved shuffle argument (for committed values) is constructed by introducing and using *randomized R1CS* constraints which are possible due to the commit-and-prove structure of Bulletproofs. Both works [BBB+18; AVY] do not claim to shuffle ElGamal ciphertext, but seem to work on Pedersen commitments, which is insufficient for many applications. Our shuffle proofs and [AVY] use the techniques from [Nef01; TW10] to avoid sorting circuits.

Unlike [BBB+18] our setting concerns (ElGamal) *encrypted* values (e.g. votes), not (Pedersen) committed values, though the choice of commitment or encryption scheme is easily modified in our construction. While it is likely that Bulletproofs can be suitably modified to cover ElGamal ciphertexts as well, e.g.

by building on their follow-up work [BAZB20], none of the works [BBB+18; AVY; BAZB20] offer an explicit solution.[2]

Lastly, we note that given any log-communication proof system, theoretically "efficient" (polynomial time) and short shuffle proofs are essentially a triviality. Just prove the relevant statements (e.g. ElGamal randomisation) in a non-black-box manner. Since we have $\mathcal{O}(\log(\mathsf{poly}(\lambda, n))) = \mathcal{O}(\log(\lambda) + \log(n))$, communication is almost logarithmic in n (and if $n \in \Omega(\lambda)$, it is logarithmic).

B.2.1. Adapting the Shuffle Argument of Bayer–Groth

The shuffle argument of Bayer and Groth [BG12] is built from two sub-arguments, a "product argument" and a "multi-exponentiation argument". A generic proof of security is given in [BG12, Theorem 5]. The former argument can be instantiated by $\mathsf{QESA}_{\mathsf{ZK}}$, or more precisely, $\mathsf{QESA}_{\mathsf{Copy}}$. The latter argument can be instantiated by $\mathsf{LMPA}_{\mathsf{ZK}}$. Since our arguments have logarithmic communication and need linearly many exponentiations, so does the resulting shuffle argument. We give a more detailed explanation below.

- CRS: $ck = (ck_Q, ck_L)$, where $ck_Q = ([\boldsymbol{g}', \boldsymbol{g}'', Q])$ is the commitment key for $\mathsf{QESA}_{\mathsf{ZK}}$ and $ck_L = [\boldsymbol{h}]$ is the commitment key for $\mathsf{LMPA}_{\mathsf{ZK}}$ (or empty if a simple zero-knowledge LMPA version is used). Here $[\boldsymbol{g}'] \in \mathbb{F}_p^n$, where $n \geq N + 2$ is a (suitably large) power of 2. Note that our commitment keys consist of random group elements.
- Common input: Old and new ciphertexts $[\mathsf{ct}_i^{\mathrm{old}}]$, $[\mathsf{ct}_i^{\mathrm{new}}] \in \mathbb{G}^2$ for $i = \{0, \ldots, N-1\}$ and ElGamal public key $[pk] \in \mathbb{G}^2$.
- Prover's witness: The random permutation $\boldsymbol{\pi} \in \{0, \ldots, N-1\}^N$ and rerandomisation randomnesses $\rho_i \in \mathbb{F}_p$ such that $[\mathsf{ct}_i^{\mathrm{new}}] = [\mathsf{ct}_{\pi_i}^{\mathrm{old}}] + \rho_i[pk]$. (Recall that $\mathsf{Enc}_{\mathsf{ElG}}(0; \rho_i) = \rho_i[pk]$ for ElGamal encryption.)
- $\mathsf{P} \to \mathsf{V}$: Compute and send the commitment $[c_{\boldsymbol{\pi}}]$ to π:

$$\begin{aligned}[c_{\boldsymbol{\pi}}] &= \mathsf{Com}_{ck_Q}(\boldsymbol{\pi}; 0, r_{\boldsymbol{\pi}}) \\ &= [g'_1 \mid g'_2, \ldots, g'_{N+1} \mid g'_{N+2}, \ldots, g'_{n-2} \mid g'_{n-1}, g'_n] \begin{pmatrix} 0 \\ \hline \boldsymbol{\pi} \\ \hline \boldsymbol{0} \\ \hline 0 \\ r_{\boldsymbol{\pi}} \end{pmatrix}\end{aligned}$$

  (Remember that $[g'_{n-1}]$ and $[g'_n]$ are reserved for randomness in $\mathsf{QESA}_{\mathsf{ZK}}$ commitments, and $[g_1]$ is reserved for the constant 1.)
- $\mathsf{V} \to \mathsf{P}$: Send $\boldsymbol{x} = (x_0, \ldots, x_{N-1}) \xleftarrow{\$} \chi_N$.
- $\mathsf{P} \to \mathsf{V}$: Send $[c_{\boldsymbol{y}}] = \mathsf{Com}_{ck_Q}(\boldsymbol{y}; 0, r_{\boldsymbol{y}})$, where $[\boldsymbol{y}] \coloneqq \boldsymbol{\pi}(\boldsymbol{x}) = (\boldsymbol{x}_{\pi_i})_i = (x_{\pi_0}, \ldots, x_{\pi_{N-1}})$.
- $\mathsf{V} \to \mathsf{P}$: Send $\zeta, z \xleftarrow{\$} \mathbb{F}_p$.
- $\mathsf{P} \leftrightarrow \mathsf{V}$: Prove following statements using (logarithmic communication) sub-protocols $\mathsf{QESA}_{\mathsf{Copy}}$ and $\mathsf{LMPA}_{\mathsf{ZK}}$:

[2] It may be tempting to argue that one part $[c_1]$ of an ElGamal ciphertext $[c_1, c_2]$ "is" a Pedersen commitment, and thus it suffices to *consider only the component* $[c_1]$. But then problem appears, that ElGamal ciphertext are only (information-theoretically) binding when the full ciphertext is considered, or the secret key is secure. In particular the component $[c_1]$ is not binding if corruption of the secret key is possible. Hence, the soundness guarantee of such a shuffle argument (which disregards $[c_2]$) seems to be useless.

- $[c_{\boldsymbol{\pi}}]$ **is a permutation and** $[c_{\boldsymbol{y}}]$ **is a commitment to** $\boldsymbol{\pi}(\boldsymbol{x})$**:** The prover shows (in zero-knowledge) that
$$\prod_{i=0}^{N-1}(\zeta\pi_i + y_i - z) = \prod_{i=0}^{N-1}(\zeta i + x_i - z).$$
Note that $\zeta[c_{\boldsymbol{\pi}}] + [c_{\boldsymbol{y}}]$ is a commitment to $\zeta\boldsymbol{\pi} + \boldsymbol{y}$, which can be used for $\mathsf{QESA}_{\mathsf{ZK}}$, or more precisely, $\mathsf{QESA}_{\mathsf{Copy}}$. Also note that the right-hand side is computable from public information.

- $[\vec{\mathrm{ct}}^{\mathrm{new}}]$ **is a rerandomised permutation of** $[\vec{\mathrm{ct}}^{\mathrm{old}}]$**:** The prover shows (in zero-knowledge) that
$$\sum_i [\mathrm{ct}_i^{\mathrm{old}}]y_i + [pk]\sum_i \rho_i x_i = \sum_i [\mathrm{ct}_i^{\mathrm{new}}]x_i.$$
This fits into our matrix multiplication proofs (with witness $\left(\begin{smallmatrix}\boldsymbol{y}\\ \boldsymbol{x}^\top\rho\end{smallmatrix}\right) \in \mathbb{F}_p^{N+1}$). Concretely, the prover commits to $\sigma := \boldsymbol{x}^\top\rho$ via $[c_\sigma] = \mathsf{Com}_{ck_Q}((\begin{smallmatrix}\boldsymbol{0}\\ \sigma\end{smallmatrix}); r_\sigma, 0) = [g'_{N+2}, g'_{n-1}](\begin{smallmatrix}\sigma\\ r_\sigma\end{smallmatrix})$ for $r_\sigma \xleftarrow{\$} \mathbb{F}_p$. He sends c_σ to the verifier and engages in a $\mathsf{LMPA}_{\mathsf{ZK}}$ protocol for
$$\left[\begin{array}{c|c|cc} g'_2,\dots,g'_{N+1} & g'_{N+2} & g'_{n-1} & g'_n \\ g'_2,\dots,g'_{N+1} & 0 & 0 & g'_n \\ \boldsymbol{0} & g'_{N+2} & g'_{n-1} & 0 \\ \mathrm{ct}_0^{\mathrm{old}}\dots\mathrm{ct}_{N-1}^{\mathrm{old}} & pk & 0 & 0 \end{array}\right]\left(\begin{array}{c} \boldsymbol{y} \\ \hline \sigma \\ \hline r_\sigma \\ r_{\boldsymbol{y}} \end{array}\right) = \begin{bmatrix} c_{\boldsymbol{y}} + c_\sigma \\ c_{\boldsymbol{y}} \\ c_\sigma \\ \boldsymbol{u} \end{bmatrix}$$
where $[\boldsymbol{u}] := \sum x_i[\mathrm{ct}_i^{\mathrm{new}}]$. The top row is added so one can run $\mathsf{LMPA}_{\mathsf{batch}}$, reducing to a $2\times n$ matrix. Since $[\boldsymbol{g}']$ has hard kernel relation, so has $[\boldsymbol{A}]$. (This is a "commitment-extension", see Remark 4.3.6.) The LMPA proof ensures in particular, that all ElGamal ciphertexts are well-formed.

Honest verifier zero-knowledge of this protocol follows from honest verifier zero-knowledge of the subprotocols. Soundness (and extraction) follows from soundness (and extraction) of the subprotocols. Namely, for fixed $\boldsymbol{\pi}$, randomly chosen $\boldsymbol{x}$ and arbitrary $\boldsymbol{y}$, the probability that $\prod_{i=0}^{N-1}(\zeta\pi_i + y_i - z) = \prod_{i=0}^{N-1}(\zeta i + x_i - z)$ holds for $\zeta, z \leftarrow \mathbb{F}_p$ if $y_i \neq x_{\pi(i)}$ is negligible thanks to the Schwartz–Zippel lemma.[3]

In [BG12], intuition and a detailed security argument is given. Despite our minor modifications, their proof adapts seamlessly to our setting. The idea of using (permutation invariant sets of) roots of polynomials to prove that one set of roots is a permutation of another goes back to [Nef01] and was extended to restricted permutations in [TW10].

A rough efficiency estimate of our scheme is $30N$ exponentiations for the prover and $10N$ exponentiations for the verifier. These are roughly twice the numbers of [BG12], when trading interaction for efficiency. However [BG12] has $O(\sqrt{N})$ size proofs, while we have $O(\log(N))$ size proofs.

B.3. Proof of Lemma 4.4.6

Lemma 4.4.6. *Let crs* $= ([\boldsymbol{g}', \boldsymbol{g}'', Q])$ *be uniformly random and as in Protocol 4.4.3 (*$\mathsf{IPA}_{\mathsf{almZK}}$*) and* $k = 2$*. Suppose that* $n \geq 4k$ *and let* $\mathbb{M}_n^+$ *be as in Definition 4.4.5. Suppose* $\boldsymbol{w}'$ *is chosen such that some component*

[3] In more detail: The degree of the (difference) polynomial in z is at most N. The two polynomials are equal if and only if they have the same roots (with multiplicity). So the sets $\{\zeta\pi_i + y_i\}_i$ and $\{\zeta i + x_i\}_i$ must be equal. The probability that $\zeta i + y = \zeta j + x$ if $i \neq j$ is negligible (for any fixed choice of x, y). Hence if the sets are equal, with overwhelming probability we find that the sets $\{(\pi_i, y_i)\}_i$ and $\{(j, x_j)\}_i$ are equal. In other words, π is a permutation of the roots. With probability $1 - \delta_{\mathrm{snd}}(\chi_N)$ all x_j are distinct, see Remark B.5.3. Hence π is a permutation of $\{0, \dots, N-1\}$.

of w'_1, w'_2, w'_{n-1}, or w'_n is uniformly random independent of $[\boldsymbol{g}'']$. Then $\mathsf{IPA}_{\mathrm{almZK}}$ *is ε-statistical HVZK with $\varepsilon = 2/p + 2(k-1)\log_2(n)/p$ for such witness distributions.*

More generally, we have the following: For arbitrary but fixed $\boldsymbol{w}'$, $\boldsymbol{w}''$, x_1, x_2, consider the combined transition and constraint matrix

$$\underbrace{\left(\begin{array}{cc} x_1\,\mathrm{id}_{n/2} & x_2\,\mathrm{id}_{n/2} \\ \boldsymbol{g}''_2 & 0 \\ 0 & \boldsymbol{g}''_1 \\ \hline \boldsymbol{w}_1'^{\top} & \boldsymbol{w}_2'^{\top} \\ \boldsymbol{r}_1'^{\top} & \boldsymbol{r}_2'^{\top} \end{array}\right)}_{=:\widetilde{\boldsymbol{M}}''} \begin{pmatrix} \boldsymbol{r}''_1 \\ \boldsymbol{r}''_2 \end{pmatrix} = \left(\begin{array}{c} \widehat{\boldsymbol{r}}' \\ u''_{-1} \\ u''_{+1} \\ \hline -\langle \boldsymbol{r}', \boldsymbol{w}'' \rangle \\ 0 \end{array}\right).$$

where $\boldsymbol{r}' \xleftarrow{\$} \mathbb{M}_n^+$. Let $\boldsymbol{M}''$ be defined by $\widetilde{\boldsymbol{M}}''$ but restricted to non-zero components of $\boldsymbol{r}''$ resp. $\widehat{\boldsymbol{r}}''$, i.e. to columns in $\mathbb{M}_n^+$ resp. in to rows in $\mathbb{M}_{n/2}$ in upper block of $\widetilde{\boldsymbol{M}}''$. Let $\boldsymbol{C}''$ be $\boldsymbol{M}''$ except that the last row $(\boldsymbol{r}_1'^{\top}, \boldsymbol{r}^{\top})$ of $\boldsymbol{M}''$ is removed. Let $\mathcal{A}$ be an unbounded HVZK adversary which picks the witnesses $(\boldsymbol{w}', \boldsymbol{w}'')$ (given $([\boldsymbol{g}', \boldsymbol{g}'', Q])$). Let δ be an upper bound on the probability that $\mathcal{A}$ chooses $(\boldsymbol{w}', \boldsymbol{w}'')$ such that $\boldsymbol{C}''$ is not surjective. Then the advantage against HVZK of $\mathcal{A}$ is at most $\delta + (1 + 2(k-1)\log(n))/p$.

Proof of Lemma 4.4.6. We restrict to $k = 2$ in the proof. We first and foremost concentrate on the subprotocol run of $\mathsf{IPA}_{\mathrm{noZK}}$ for $\langle \beta\boldsymbol{w}' + \boldsymbol{r}', \beta\boldsymbol{w}'' + \boldsymbol{r}'' \rangle$. We analyse the rounds of this subprotocol, similar to Lemma 4.3.23. That is, we consider the transition matrix and the probability for it to be surjective. as in Lemma 4.3.23 surjectivity ensures that $[u_{\pm 1}]$ are uniform. We will analyze the transition matrices for $\boldsymbol{r}'$ and $\boldsymbol{r}''$ separately; this only strengthens our claim, since if $[u''_\ell]$ is uniform, then clearly $[u_\ell] = [u'_\ell] + [u''_\ell]$ is. For $\boldsymbol{r}'$, we need not analyze anything, except in the round where $n = k$ Thus, we concentrate on $\boldsymbol{r}''$ and its constraints. As in Lemma 4.3.23, we treat the execution of the masked protocol as a linear combination, namely, in each round we reduce from $\beta\boldsymbol{w}' + \boldsymbol{r}'$ to $\beta\widehat{\boldsymbol{w}}' + \widehat{\boldsymbol{r}}'$ and likewise for $\boldsymbol{r}''$.

The First Round. Recall that, if $\boldsymbol{M} \in \mathbb{F}_p^n \to \mathbb{F}_p^m$ is surjective, then the image a uniformly drawn element is uniformly distributed. The formal claim and proof this are given in Lemma B.4.4 and trivially generalize to surjective linear maps in the setting where some outputs are prescribed, i.e. where some rows are used for constraints (as in our case). As such, we aim to show surjectivity of $\boldsymbol{M}$. By definition, the matrix $\widetilde{\boldsymbol{M}}'$ (resp. $\widetilde{\boldsymbol{M}}''$) is constructed essentially as in Lemma B.4.5:

- The top block is the transition matrix for a run of $\mathsf{LMPA}_{\mathrm{noZK}}$. (Recall that $\mathsf{IPA}_{\mathrm{noZK}}$ is sends messages as a linear combination two such runs, one for $\boldsymbol{r}'$ and one for $\boldsymbol{r}''$.)
- The two bottom rows correspond to the constraints $\langle \boldsymbol{r}', \boldsymbol{r}'' \rangle = 0$ and $\langle \boldsymbol{w}', \boldsymbol{r}'' \rangle = -\langle \boldsymbol{r}', \boldsymbol{w}'' \rangle$

Removing the zero-rows (w.r.t. $\widehat{\boldsymbol{r}}$) and columns (w.r.t. $\boldsymbol{r}$) leads to

$$\underbrace{\left(\begin{array}{ccccc} x_1\,\mathrm{id}_2 & 0 & 0 & x_2\,\mathrm{id}_2 & 0 \\ 0 & x_1\,\mathrm{id}_{\dim(\mathbb{M}_{n/4})-2} & 0 & 0 & 0 \\ 0 & 0 & x_1\,\mathrm{id}_2 & 0 & x_2\,\mathrm{id}_2 \\ \boldsymbol{g}''_{n/2+1,n/2+2} & \boldsymbol{g}''_{n/2+\mathbb{M}_{n/4}\setminus\{1,2\}} & \boldsymbol{g}''_{n-1,n} & 0 & 0 \\ 0 & 0 & 0 & \boldsymbol{g}_{n/2+1,n/2+2} & \boldsymbol{g}''_{n/2-1,n/2} \\ \hline \boldsymbol{w}'_{1,2} & \boldsymbol{w}'_{\mathbb{M}_{n/4}\setminus\{1,2\}} & \boldsymbol{w}'_{n/2-1,n/2} & \boldsymbol{w}'_{n/2+1,n/2+2} & \boldsymbol{w}'_{n-1,n} \\ \boldsymbol{r}'_{1,2} & \boldsymbol{r}'_{\mathbb{M}_{n/4}\setminus\{1,2\}} & \boldsymbol{r}'_{n/2-1,n/2} & \boldsymbol{r}'_{n/2+1,n/2+2} & \boldsymbol{r}'_{n-1,n} \end{array}\right)}_{=:\boldsymbol{M}} \begin{pmatrix} \boldsymbol{r}''_{1,2} \\ \boldsymbol{r}''_{\mathbb{M}_{n/4}\setminus\{1,2\}} \\ \boldsymbol{r}''_{n/2-1,n/2} \\ \boldsymbol{r}''_{n/2+1,n/2+2} \\ \boldsymbol{r}''_{n-1,n} \end{pmatrix} = \left(\begin{array}{c} \widehat{\boldsymbol{r}}''_{1,2} \\ \widehat{\boldsymbol{r}}''_{\mathbb{M}_{n/4}\setminus\{1,2\}} \\ \widehat{\boldsymbol{r}}''_{n/2-1,n/2} \\ u''_{-1} \\ u''_1 \\ \hline -\langle \boldsymbol{r}', \boldsymbol{w}' \rangle \\ 0 \end{array}\right)$$

where the subscripts indicate which component the components of the vector slices. Note that $\mathbb{M}_{n/4} = \mathbb{M}_{n/2} \setminus \{n/2 - 1, n/2\}$. Clearly, the block with $x_1\,\mathrm{id}_{\mathbb{M}_{n/4}\setminus\{1,2\}}$ can be eliminated. The analysis of the result is essentially identical to the analysis of the case $n = 8$, for which the transition matrix is

$$\boldsymbol{M} := \left(\begin{array}{cccc|cccc} x_1 & & & & x_2 & & & \\ & x_1 & & & & x_2 & & \\ & & x_1 & & & & x_2 & \\ & & & x_1 & & & & x_2 \\ g_5 & g_6 & g_7 & g_8 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & g_1 & g_2 & g_3 & g_4 \\ \hline w_1 & w_2 & w_3 & w_4 & w_4 & w_6 & w_7 & w_8 \\ r_1 & r'_2 & r'_3 & r'_4 & r'_4 & r'_6 & r'_7 & r'_8 \end{array}\right)$$

The analysis of surjectivity of $\boldsymbol{M}''$ is complicated by the fact that $\boldsymbol{w}$ may depend on $[\boldsymbol{g}', \boldsymbol{g}'', Q]$. If we simply assume that $\mathcal{A}$ picks surjective $\boldsymbol{C}''$ (which is $\boldsymbol{M}''$ with the last row removed) except with probability δ, then $\boldsymbol{M}''$ is surjective except with probability $\delta + 1/p$ (due to $\boldsymbol{r}'$ being uniform in $\mathbb{M}_n^+$). This is the easy case.

Now suppose that (by assumption) some w_i is uniformly random and *stochastically independent* of the other w_i. W.l.o.g. the other w_j $(j \neq i)$ are a deterministic function f of $\boldsymbol{g}$ and $x_1, \dots, x_k$. (Dealing with a probabilistic f works by conditioning on its randomness.) We compute the determinant by Lagrange expansion. Suppose for concreteness, that w_8 is the uniformly random term. Then the determinant is of the form

$$\det(\boldsymbol{M}) = w_8 \cdot \underbrace{\det\begin{pmatrix} x_1 & & & & x_2 & & \\ & x_1 & & & & x_2 & \\ & & x_1 & & & & x_2 \\ & & & x_1 & & & \\ g_5 & g_6 & g_7 & g_8 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & g_1 & g_2 & g_3 \\ r_1 & r'_2 & r'_3 & r'_4 & r'_4 & r'_6 & r'_7 \end{pmatrix}}_{=:\eta} + \mathsf{poly}(\boldsymbol{g}, x_1, x_2, w_1, \dots, w_7)$$

where poly is a polynomial in the given variables (which notably exclude w_8). It is easy to see that $\Pr[\eta = 0] \leq 3/p$, e.g. by further expanding r'_7, g_5 and g_6 and using that $x_1, x_2 \neq 0$ by construction. Importantly, w_8 and η are *stochastically independent*; here we exploit HVZK to ensure that x_1, x_2 are independent of w_8. Thus, except with probability $3/p$, $\det(\boldsymbol{M})$ is a non-zero linear polynomial in w_8, hence except with probability $4/p$ the determinant $\det(\boldsymbol{M})$ is non-zero and thus $\boldsymbol{M}$ is surjective. Dealing with the case where another w_i is the independently uniform one is completely analogous.

Finally observe, that $(\widehat{\boldsymbol{r}}', \widehat{\boldsymbol{r}}'', u''_{-1}, u''_1)$ has statistical distance of at most $4/p$ from uniformly random, since it is uniformly distributed whenever $\boldsymbol{M}''$ is surjective. Also recall that without the constraints, the distance would have been $2/p$ instead of $4/p$. This additional term $2/p$ appears the lemma's claim, and it is the main difference to Lemma 4.3.23.

Recursive Rounds. From this point on, we are working with the same masking set $\mathbb{M}_{n/2}$ as in Lemma 4.3.23. Moreover, there are no more constraints placed on $\boldsymbol{r}''$. Thus, the "transition matrix" $\boldsymbol{M}''$ for $\boldsymbol{r}''$ is completely analogous to the one in Lemma 4.3.23, and hence each $[u''_\ell]$ is uniformly random unless $\boldsymbol{M}''$ is not surjective. Clearly, if $[u''_\ell]$ is uniformly random, so is $[u_\ell]$. As in Lemma 4.3.23, the probability that $\boldsymbol{M}''$ is not surjective (assuming uniform $\boldsymbol{g}''$ and $\boldsymbol{r}''$) is at most $2(k-1)/p$ per recursion.

The Last Round. Finally, consider the last recursive round, i.e. the reduction from $n = k$ to $n = 1$. Since by assumption, we started with $n > k$ (hence there are no constraints on $\boldsymbol{r}'$ and $\boldsymbol{r}''$), and since $\mathbb{M}_k = \{1, \dots, k\}$, in this case both $\boldsymbol{r}'$ and $\boldsymbol{r}''$ are uniformly random masks for $\boldsymbol{w}'$ and $\boldsymbol{w}''$. Thus, even $\beta\boldsymbol{w}' + \boldsymbol{r}'$ and $\beta\boldsymbol{w}'' + \boldsymbol{r}''$ reveals nothing about the original witness $(\boldsymbol{w}', \boldsymbol{w}'')$.

HVZK Simulation. Simulation works as follows: The simulator chooses random $\boldsymbol{w}', \boldsymbol{w}''$ for round $n = k$, and honestly computes the response. The previous rounds are simulated in reverse, as usual. Namely, by picking $[u_{\pm 1}]$ uniformly at random and computing $[u_0]$ from this and the challenge. Eventually, the simulator reaches $[c] = [u_0]$ and computes the unique $[c_{\boldsymbol{r}}] = [c] - (\beta[c_{\boldsymbol{w}}] + \beta^2[Q]t)$. As in Lemma 4.3.23, each recursive round accumulates (via union-bound) an error of $2(k-1)/p$, except for the first round, which contributes $4(k-1)/p$, and the last round ($n = k$), which contributes no error (since the witness is already fully masked). Overall, the statistical distance is at most $\varepsilon = 2/p + 2(k-1)\log_2(n)/p$. For the more general case where $\mathcal{A}$ chooses $\boldsymbol{w}$, the statistical distance is at most $\varepsilon = \delta + 1/p + 2(k-1)\log_2(n)/p$. □

B.4. $\mathsf{LMPA}_{\mathsf{ZK}}$ for general $[A]$

In this section, we provide the details on how to add zero-knowledge to $\mathsf{LMPA}_{\mathsf{noZK}}$ for general adversarial $[\boldsymbol{A}]$ following the outline in Section 4.3.5.3. Our proof system separates the masking randomness from the actual witness and is a linear combination of multiple protocol instances of $\mathsf{LMPA}_{\mathsf{noZK}}$, namely: The actual protocol for $[\boldsymbol{A}] =: [\boldsymbol{H}^{(0)}]$, and protocols for $[\boldsymbol{H}^{(i)}]$, $i = 1, \dots, m$, where $[\boldsymbol{H}^{(i)}]$ essentially contains a Pedersen commitment key in the i-th row and is zero otherwise.

To keep things simple, we let $m = 1, k = 2$ in the following discussion. Intuitively, we want to run a "randomness-extended" protocol for $[\boldsymbol{B}] = [\boldsymbol{A}|\boldsymbol{H}]\binom{\boldsymbol{w}}{\boldsymbol{r}}$. The intuition is that $\boldsymbol{r}$ will randomise all $[u_{\pm 1}]$'s (because $[\boldsymbol{H}]$ is not adversarial). Unfortunately, this intuition is wrong: $[u_1] = [\boldsymbol{H}]\boldsymbol{w}$ is certainly not zero-knowledge. The problem is how $\mathsf{LMPA}_{\mathsf{noZK}}$ divides the statement. Appropriate shuffling of $[\boldsymbol{B}]$ and $\binom{\boldsymbol{w}}{\boldsymbol{r}}$ would solve this. Instead, we work with a linear combination of $\mathsf{LMPA}_{\mathsf{noZK}}$ instances.

More precisely, we run *two* arguments, one for $[\boldsymbol{A}]\boldsymbol{w} = [\boldsymbol{t}']$ and one for $[\boldsymbol{H}]\boldsymbol{r} = [\boldsymbol{t}'']$. The messages $[u_{-1}]$ and $[u_1]$ are the sums of the messages which individual protocols would send, e.g. $[u_{-1}] = [\boldsymbol{A}_2]\boldsymbol{w}_1 + [\boldsymbol{H}_2]\boldsymbol{r}_1$. Concretely

$$\begin{bmatrix} u'_{-1} \\ u'_1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{A}_2\boldsymbol{w}_1 \\ \boldsymbol{A}_1\boldsymbol{w}_2 \end{bmatrix}, \qquad \begin{bmatrix} u''_{-1} \\ u''_1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{H}_2\boldsymbol{r}_1 \\ \boldsymbol{H}_1\boldsymbol{r}_2 \end{bmatrix}, \qquad \begin{bmatrix} u_{-1} \\ u_1 \end{bmatrix} = \begin{bmatrix} u'_{-1} \\ u'_1 \end{bmatrix} + \begin{bmatrix} u''_{-1} \\ u''_1 \end{bmatrix}$$

This ensures that the $[u''_{\pm 1}]$ are uniformly random in every round, because $[u''_{\pm 1}]$ is. In the base case of the recursion, i.e. small n, the prover proves $[\boldsymbol{A}]\boldsymbol{w} + [\boldsymbol{H}]\boldsymbol{r} = [\boldsymbol{t}]$ in zero-knowledge, using (for concreteness) Protocol Σ_{std}.

To keep our protocol modular, we split it into two steps. In the first step, we only introduce masking to achieve HVZK at the price of relaxed soundness. In the second step, we strengthen soundness again.

Protocol B.4.1 ($\mathsf{LMPA}_{\mathsf{almSnd}}$). The following is a protocol to prove $\exists \boldsymbol{w}\colon [\boldsymbol{t}^{(0)}] = [\boldsymbol{A}]\boldsymbol{w}$. Common input is $([\boldsymbol{A}], [\boldsymbol{t}^{(0)}]) \in \mathbb{G}^{m \times n} \times \mathbb{G}^n$ and some $[\boldsymbol{h}] \in \mathbb{G}^n$ (typically derived from the CRS when this protocol is used as a subprotocol). We assume $n = k^\ell$. Moreover, we let $[\boldsymbol{H}^{(i)}] \in \mathbb{G}^{m \times n}$ for $i = 1, \dots, m$, be defined as the matrix with $[\boldsymbol{h}]$ in the i-th row and zeroes elsewhere, i.e. $[\boldsymbol{H}^{(i)}] = \boldsymbol{e}_i[\boldsymbol{h}]^\top$. We use a superscript 0 for terms related to $[\boldsymbol{A}]$, e.g. $[\boldsymbol{H}^{(0)}] := [\boldsymbol{A}]$. The prover's witness is some $\boldsymbol{w} \in \mathbb{F}_p^n$, also written $\boldsymbol{r}^{(0)}$, with $[\boldsymbol{A}]\boldsymbol{r}^{(0)} = [\boldsymbol{t}^{(0)}]$.

- P $\longrightarrow$ V: (Step 1: Prepare masking.) Pick $\boldsymbol{r}^{(i)} \xleftarrow{\$} \mathbb{M}_n \le \mathbb{F}_p^n$ and compute $[\boldsymbol{t}^{(i)}] = [\boldsymbol{H}^{(i)}]\boldsymbol{r}^{(i)}$. Send $[\boldsymbol{t}^{(i)}]$ for $i = 1, \dots, m$.
- Both prover and verifier now consider the effective statement

$$[\boldsymbol{A}|\boldsymbol{H}^{(1)}|\dots|\boldsymbol{H}^{(m)}]\begin{pmatrix}\boldsymbol{w}\\ \boldsymbol{r}^{(1)}\\ \vdots\end{pmatrix} = [\boldsymbol{t}] \coloneqq \sum_{i=0}^{m}[\boldsymbol{t}^{(i)}].$$

- P $\longrightarrow$ V: (Step 2: Shrinking AoK.) Let $[\boldsymbol{H}^{(i)}] = [\boldsymbol{H}_1^{(i)}|\dots|\boldsymbol{H}_k^{(i)}]$ with $\boldsymbol{H}_j^{(i)} \in \mathbb{G}^{m\times n/k}$. Compute $[\boldsymbol{u}_\ell] = \sum_{i=0}^{m}[\boldsymbol{u}_\ell^{(i)}]$, where $[\boldsymbol{u}_\ell^{(i)}]$ is computed as usual, i.e. $[\boldsymbol{u}_\ell^{(i)}] = \sum_{j-i=\ell}[\boldsymbol{H}_\ell^{(i)}]\boldsymbol{r}_\ell^{(i)}$. Send $[\boldsymbol{u}_\ell]$ for $\ell = \pm 1, \dots, \pm(k-1)$.
- V $\longrightarrow$ P: Pick $\boldsymbol{z} \xleftarrow{\$} \widetilde{\chi}_{2k-1}^{\mathrm{mon}}$ (with associated $\boldsymbol{x}, \boldsymbol{y}$). Send $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$.
- P $\longrightarrow$ V: As in $\mathsf{LMPA}_{\mathrm{noZK}}$, compute $\boldsymbol{w} = \boldsymbol{x}^\top\vec{\boldsymbol{w}} = \sum_j x_j\boldsymbol{w}_i$ and $\widehat{\boldsymbol{r}}^{(i)} = \boldsymbol{x}^\top\vec{\boldsymbol{r}}^{(i)} = \sum_j x_j\boldsymbol{r}_j^{(i)}$ and $[\widehat{\boldsymbol{A}}] = \boldsymbol{x}^\top[\vec{\boldsymbol{A}}] = \sum_j x_j[\boldsymbol{A}_j]$, $[\widehat{\boldsymbol{H}}^{(i)}] = \sum_j x_j[\boldsymbol{H}_j^{(i)}]$, and $[\widehat{\boldsymbol{t}}] = \boldsymbol{z}^\top\vec{\boldsymbol{u}} = \sum_\ell z_j\boldsymbol{u}_\ell$, for the reduced statement (which V also computes).
  If $n > k$, engage recursively in the AoK for this statement, i.e. goto Step 3. If $n \le k$, engage in (for concreteness) Protocol Σ_{std} to prove the statement.

It is easy to check that Protocol B.4.1 is complete.

Remark B.4.2. Suppose that $[\boldsymbol{h}]$ is chosen such that $[\boldsymbol{h}]$ is zero outside of $\mathbb{M}_n$. (An honest prover would never use some $\boldsymbol{r}^{(i)}$ which is non-zero outside $\mathbb{M}_n$, so this is a sensible assumption.) Then the overhead in exponentiations and randomness introduced by $\mathsf{LMPA}_{\mathrm{almSnd}}$ compared to $\mathsf{LMPA}_{\mathrm{noZK}}$ is roughly $2m\#\mathbb{M}_n \approx 4mk\log_k(n)$, hence $\mathcal{O}(m\log_k(n))$ which can be much less than full masking $\mathcal{O}(n)$.

Lemma B.4.3. *Protocol* $\mathsf{LMPA}_{\mathrm{almSnd}}$ *has* $(2k-1, \dots, 2k-1, 2)$*-special (relaxed) soundness for finding a preimage* $\vec{\boldsymbol{v}} \in (\mathbb{F}_p^n)^m$ *with* $[\boldsymbol{A}|\boldsymbol{H}^{(1)}|\dots|\boldsymbol{H}^{(m)}]\left(\begin{smallmatrix}\boldsymbol{v}_0\\ \cdots\\ \boldsymbol{v}_m\end{smallmatrix}\right) = [\boldsymbol{t}^{(0)}]$*, or a non-trivial kernel element of* $[\boldsymbol{A}|\boldsymbol{H}'^{(1)}|\dots|\boldsymbol{H}'^{(m)}]$*. Here,* $[\boldsymbol{H}'^{(i)}]$ *consists only of the non-zero components of* $[\boldsymbol{H}^{(i)}]$*. (It is easy to find non-trivial kernel elements if* $[\boldsymbol{h}]$ *has zeroes, so we exclude them, cf. Remark 4.2.2.)*

Moreover, $\mathsf{LMPA}_{\mathrm{almSnd}}$ *has* $(k, \dots, k, 2)$*-quick* $(2k, \dots, 2k, 2)$*-short extractability for the same relation.*

Note Lemma B.4.3 *does not assert* a witness $\boldsymbol{w} \in \mathbb{F}_p^n$ for $[\boldsymbol{A}]\boldsymbol{w} = [\boldsymbol{t}^{(0)}]$. That will be assured by an extra step in a later protocol.

Proof. We only sketch the proof. Let *tree* be a valid μ-tree of transcripts. First of all, we can extract the base subprotocol of Step 3. Using these witnesses, we can extract the linearly combined argument essentially as in Lemma 4.3.19.[4] There is only a minor technicality: $\boldsymbol{H}^{(i)}$ has $\mathbf{0}$-columns and, a priori, an extracted $\boldsymbol{v}_i$ may be non-zero in such a column. However, w.l.o.g. $\boldsymbol{v}_i$ is zero in any $\mathbf{0}$-column of $\boldsymbol{H}^{(i)}$, as this does not affect the result. Thus, the argument is indeed completely analogous. As the submatrix $[\boldsymbol{H}'^{(i)}]$ in Lemma B.4.3 is defined by “omitting” the irrelevant $\mathbf{0}$-components, the claim follows. □

[4] Indeed, after suitably permuting the columns of $[\boldsymbol{A}|\boldsymbol{H}^{(1)}|\dots|\boldsymbol{H}^{(m)}]$, witness, and randomness, the exact same reasoning as in Lemma 4.3.19 works for the recursive step.

To prove HVZK for Protocol $\mathsf{LMPA}_{\text{almSnd}}$, we first show that the prover's messages $[\boldsymbol{u}_\ell]$ in the recursive steps are almost always uniformly distributed. This yields statistical HVZK via straightforward simulation.

As a preparation, note following (easy) linear algebra facts:

Lemma B.4.4. *Let $\mathbb{V} \cong \mathbb{F}_p^n$ and $\mathbb{W} \cong \mathbb{F}_p^m$ be some vector spaces. Let $\boldsymbol{L}\colon \mathbb{V} \longrightarrow \mathbb{W}$ be a linear map (i.e. a matrix $\boldsymbol{L} \in \mathbb{F}_p^{m\times n}$). Then $\boldsymbol{r} \mapsto \boldsymbol{L}\boldsymbol{r}$ for $\boldsymbol{r} \xleftarrow{\$} \mathbb{V}$ uniformly random induces the uniform distribution on $\mathbb{W}$ if and only if $\boldsymbol{L}$ is surjective. (Equivalently, if the rows of $\boldsymbol{L}$ (as a matrix) are linearly independent.)*

Proof. Every $y \in \operatorname{im}(\boldsymbol{L})$ has the same number of preimages, namely $P = \#\ker(\boldsymbol{L})$. Thus, if $\boldsymbol{u} \in \mathbb{V}$ is uniformly distributed, then for every $y \in \operatorname{im}(\boldsymbol{L})$ we have $\Pr[\boldsymbol{L}\boldsymbol{u} = \boldsymbol{y}] = P/\#\mathbb{V}$. Hence, if $\boldsymbol{L}$ is surjective, then $\operatorname{im}(\boldsymbol{L}) = \mathbb{W}$, $P = \#\mathbb{V}/\#\mathbb{W}$ and thus $\Pr[\boldsymbol{L}\boldsymbol{u} = \boldsymbol{y}] = 1/\#\mathbb{W} = p^{-m}$, as claimed. □

Lemma B.4.5. *Consider Protocol B.4.1 ($\mathsf{LMPA}_{\text{almSnd}}$). Suppose that (at least) all components of $[\boldsymbol{h}]$ in $\mathbb{M}_n$ are distributed uniformly random (and the rest may be 0).*

Denote by $\boldsymbol{U}$ the list consisting of messages $[\boldsymbol{u}_\ell]$ of all *recursive rounds of an honest execution. Then, $\boldsymbol{U}$ is ε-close to uniformly random for $\varepsilon \le 2m(k-1)\log_k(n)/p$.*

The lemma is proven almost exactly like the simpler case where $k = 2$, $m = 1$ in Lemma 4.3.23. Namely, the different rows of $[\boldsymbol{A}]$ are masked by the different $[\boldsymbol{H}^{(i)}]$ in a completely independent manner, except that "randomization failures" can now happen in every row, which explains the factor m in the statistical HVZK error. Moreover, in Lemma 4.3.23 we implicitly used that the masking behaved like a linear combination, so that we could reduce to studying $[\boldsymbol{g}]\boldsymbol{r}$. In the current case, we handle a linear combination by definition. Overall, the one major difference in the analysis is that we take care of the case $k > 2$, which is more technical. Another difference is that we do not mask $\boldsymbol{w}$ directly (i.e. the protocol does not run on $\beta\boldsymbol{w} + \boldsymbol{r}$, but instead $\boldsymbol{w}$ is extended to $\left(\begin{smallmatrix}\boldsymbol{w}\\ \boldsymbol{r}\end{smallmatrix}\right)$). This affects the handling of the base case for $k = 2$, which now uses Σ_{std}.

As in Lemma 4.3.23, in the proof we will consider the "transition maps", prove that they are surjective (with high probability), and then by induction derive that (with high probability) the total transcript $[\boldsymbol{U}]$ of masks is uniformly random. The proof follows.

Proof. It suffices to consider $m = 1$, because the matrices $\boldsymbol{H}^{(i)}$ are constructed such that they mask the i-th row only (they are zero in all other rows). Evidently, there is also no "interference" between rows in the protocol because $\widehat{\boldsymbol{H}}^{(i)}$ is again only non-zero in the i-th row ($i \neq 0$). However, as the transition matrix can fail to be surjective in each row separately, we must compensate for this in the final statistical error, which we do by a union bound. Consequently, we now consider $[\boldsymbol{A}], [\boldsymbol{H}] \in \mathbb{G}^{1\times n}$ and drop the superscripts.

As a first step, we consider the case where the masking randomness is simply taken from $\mathbb{F}_p^n$ uniformly, i.e. $\mathbb{M}_n = \{1, \dots, n\}$. By construction, we have (for $s = n/k$)

$$\underbrace{\left(\begin{array}{ccccc} x_1 \operatorname{id}_s & x_2 \operatorname{id}_s & \dots & x_{k-1} \operatorname{id}_s & x_k \operatorname{id}_s \\ \boldsymbol{H}_k & 0 & \dots & 0 & 0 \\ \boldsymbol{H}_{k-1} & \boldsymbol{H}_k & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \hline \boldsymbol{H}_1 & \boldsymbol{H}_2 & \dots & \boldsymbol{H}_{k-1} & \boldsymbol{H}_k \\ \hline \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & \boldsymbol{H}_1 & \boldsymbol{H}_2 \\ 0 & 0 & \dots & 0 & \boldsymbol{H}_1 \end{array}\right)}_{=:L'} \begin{pmatrix} \boldsymbol{r}_1 \\ \boldsymbol{r}_2 \\ \vdots \\ \boldsymbol{r}_{k-1} \\ \boldsymbol{r}_k \end{pmatrix} = \left(\begin{array}{c} \widehat{\boldsymbol{r}} \\ u_{1-k} \\ u_{2-k} \\ \vdots \\ \hline u_0 \\ \hline \vdots \\ u_{k-2} \\ u_{k-1} \end{array}\right) \tag{B.4.1}$$

Let $\boldsymbol{L}$ be the matrix where the row $(\boldsymbol{H}_1, \dots, \boldsymbol{H}_k)$ corresponding to $\boldsymbol{u}_0$ is removed from $\boldsymbol{L}$. If $\boldsymbol{r}_1, \dots, \boldsymbol{r}_k$ are uniformly distributed, then by Lemma B.4.4 the matrix-vector product

$$\boldsymbol{L} \begin{pmatrix} \boldsymbol{r}_1 \\ \boldsymbol{r}_2 \\ \vdots \\ \boldsymbol{r}_{k-1} \\ \boldsymbol{r}_k \end{pmatrix} = \begin{pmatrix} \widehat{\boldsymbol{r}} \\ u_{1-k} \\ u_{2-k} \\ \vdots \\ u_{-1} \\ u_1 \\ \vdots \\ u_{k-2} \\ u_{k-1} \end{pmatrix} \tag{B.4.2}$$

is also uniformly distributed if $\boldsymbol{L}$ is surjective. However, this depends on the $\boldsymbol{H}_i$. We (will) see:

- If $\boldsymbol{H}$ is uniformly random, then with high probability $\boldsymbol{L}$ is surjective, i.e. has independent rows.
- If $\boldsymbol{L}$ is surjective, then u_ℓ (for $\ell \neq 0$) and $\widehat{\boldsymbol{r}}$ are jointly uniformly distributed.
- The property is "preserved by reduction", i.e. even after "application" of $\boldsymbol{x}$ to $\boldsymbol{H}$, $\widehat{\boldsymbol{H}}$ is uniformly distributed again, and the transition matrix for $\widehat{\boldsymbol{H}}$ is again surjective (with high probability), and so on.

Claim B.4.6. *If the $\boldsymbol{H}_i$s are uniformly random (of dimension $s \geq 2$), then except with probability probability at most $2(k-1)/p$, the matrix $\boldsymbol{L}$ is invertible.*

Proof. One quick way to see this is following observation: Let f be the determinant of a square submatrix $\boldsymbol{L}$, which is a polynomial in the coefficient of $\boldsymbol{H}$. If f is not constantly 0, then by Schwartz–Zippel, with probability at least $\deg(f)/p$, we have $f(\boldsymbol{H}) \neq 0$, i.e. $\boldsymbol{L}$ is bijective, in particular surjective. Thus, it suffices to find (for fixed $\boldsymbol{x}$) a single choice of $\boldsymbol{H}$ such that $\boldsymbol{L}$ is invertible. The claim follows since $\deg(f) = 2(k-1)$.

We argue the case of $s = 2$ in more detail. If $s = 2$, then $\boldsymbol{L}$ is already square. Thus, it suffices to see that $\det(\boldsymbol{L})$ is a non-zero polynomial in $\boldsymbol{H}$ of degree at most $2(k-1)$. The degree bound is automatic, since there are only $2(k-1)$ rows with coefficient in $\boldsymbol{H}$ (and $s = 2$ rows with x_is, which are arbitrary but fixed and non-zero). Thus, it suffices to see that $\boldsymbol{L}$ is non-zero. For this, observe that there is only a single choice of rows and column for Laplace expansion which leads to a coefficient of $H_{k,1}^{k-1} H_{1,1}^{k-1}$, namely, the one which results in $H_{k,1}^{k-1} H_{1,1}^{k-1} x_1 x_k$ which is a non-zero monomial. Hence, indeed, $\det(\boldsymbol{L})$ is a non-zero polynomial in $\boldsymbol{H}$ and the claim follows. (We note that, to apply the Schwartz–Zippel lemma, we also use that the x_i are stochastically independent from $H_{k,1}$, $H_{1,1}$. This holds since we consider HVZK.) □

Now, let us consider the case where $\mathbb{M}_n$ is as in Definition 4.3.21. Recall that $\mathbb{M}_n$ has following structural properties (for general k):

- $\mathbb{M}_{k^\ell,i} := \mathbb{M}_{k^\ell} \cap \{ik^{\ell-1}, \dots, (i+1)k^{\ell-1} - 1\}$ satisfies $\mathbb{M}_{n,1} = \mathbb{M}_{n/k}$ and $\mathbb{M}_{k^\ell,i} = \{ik^{\ell-1} - 1, ik^{\ell-1}\}$.
- Split $\boldsymbol{r} \in \mathbb{F}_p^{k^\ell}$ into k pieces $\boldsymbol{r}_i$ as usual, and define $\widehat{\boldsymbol{r}}$ as usual. Then the components in $\{k^{\ell-1}-1, k^{\ell-1}\}$ of each $\boldsymbol{r}_i$ are linearly combined in $\widehat{\boldsymbol{r}}$.

Observe that, $\widehat{\boldsymbol{r}}$ lies in the "correct" subspace of $\mathbb{F}_p^{n/k}$, namely in $\mathbb{M}_{n/k}$. We will show that, the distribution of $(\widehat{\boldsymbol{r}}, [u_{\pm 1}], \dots, [u_{\pm(k-1)}])$ is uniformly random in $\mathbb{M}_{n/k} \times \mathbb{G}^{2(k-1)}$.

To this end, we now analyze the surjectivity of transition matrix $\boldsymbol{L}$. First of all, we remove the columns not in $\mathbb{M}_n$ from $\boldsymbol{L}$. (The respective columns are "useless" for randomisation, since $\boldsymbol{r}$ (and $\boldsymbol{H}$) is only non-zero for components in $\mathbb{M}_n$.) Second, we remove the rows not in $\mathbb{M}_{n/k}$ from the upper part of $\boldsymbol{L}$ (corresponding to $\widehat{\boldsymbol{r}}$). (Again, since $\widehat{\boldsymbol{r}}$ need only be non-zero for components in $\mathbb{M}_{n/k}$, we only need surjectivity in those components.) Note that now all remaining components of $\boldsymbol{r}$ and $\boldsymbol{H}$ were chosen uniformly at random (as part of the induction hypothesis).

We denote this submatrix of the relevant parts of $\boldsymbol{L}$ by $\boldsymbol{M}$. Note that $\boldsymbol{M}$ has dimensions $\dim(\mathbb{M}_n) \times (\dim(\mathbb{M}_{n/k}) + 2(k-1))$, where $\dim \mathbb{M}_{n/k}$ stems from $\widehat{\boldsymbol{r}}$ and $2(k-1)$ from $[u_{\pm i}]$ (for $i = 1, \dots, k-1$). Since $\dim(\mathbb{M}_n) = \dim(\mathbb{M}_{n/k}) + 2(k-1)$, we see that $\boldsymbol{M}$ is in fact a square matrix.

As in Lemma 4.3.23, we now split $\boldsymbol{r}_1$ into $(\boldsymbol{r}'_1, \boldsymbol{r}''_1)$, where $\boldsymbol{r}'_1$ corresponds to components $\mathbb{M}_{n/k} \setminus \{k^{\ell-1} - 1, k^{\ell-1}\}$ and $\boldsymbol{r}''_1$ corresponds to components $\{k^{\ell-1} - 1, k^{\ell-1}\}$. Similarly, we define $\boldsymbol{r}''_i$ corresponding to the components $\{k^{\ell-1} - 1, k^{\ell-1}\}$ in $\boldsymbol{r}_i$. We also define $\widehat{\boldsymbol{r}}'$, $\widehat{\boldsymbol{r}}''$ and $\boldsymbol{H}'_i$, $\boldsymbol{H}''_i$ analogously. Thus, we only consider components which are not always zeroed (by an honest prover). Note that we exploited the structure of $\mathbb{M}_n$ here. Now, the transition during a recursive step is the following:

$$\underbrace{\left(\begin{array}{c|ccccc}
x_1 \,\mathrm{id}_{\dim(\mathbb{M}_{n/k})-2} & 0 & 0 & \dots & 0 & 0 \\ \hline
0 & x_1 \,\mathrm{id}_2 & x_2 \,\mathrm{id}_2 & \dots & x_{k-1} \,\mathrm{id}_2 & x_k \,\mathrm{id}_2 \\
\boldsymbol{H}'_k & \boldsymbol{H}''_k & 0 & \dots & 0 & 0 \\
\boldsymbol{H}'_{k-1} & \boldsymbol{H}''_{k-1} & \boldsymbol{H}''_k & \dots & 0 & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\boldsymbol{H}'_2 & \boldsymbol{H}''_2 & \boldsymbol{H}''_3 & \dots & \boldsymbol{H}''_k & 0 \\
0 & 0 & \boldsymbol{H}''_1 & \dots & \boldsymbol{H}''_{k-2} & \boldsymbol{H}''_{k-1} \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
0 & 0 & \dots & \dots & \boldsymbol{H}''_1 & \boldsymbol{H}''_2 \\
0 & 0 & \dots & \dots & 0 & \boldsymbol{H}''_1
\end{array}\right)}_{=:\boldsymbol{M}}
\begin{pmatrix} \boldsymbol{r}'_1 \\ \boldsymbol{r}'' \\ \boldsymbol{r}''_2 \\ \vdots \\ \boldsymbol{r}''_{k-1} \\ \boldsymbol{r}''_k \end{pmatrix}
=
\left(\begin{array}{c} \widehat{\boldsymbol{r}}' \\ \hline \widehat{\boldsymbol{r}}'' \\ u_{1-k} \\ u_{2-k} \\ \vdots \\ u_{-1} \\ u_1 \\ \vdots \\ u_{k-2} \\ u_{k-1} \end{array}\right) \tag{B.4.3}$$

To prove surjectivity of this transition matrix, we can remove the left block with $\mathrm{id}_{\dim(\mathbb{M}_{n/k})-2}$. What remains is a matrix which looks exactly like the one we dealt with in Claim B.4.6 for $s = 2$. Thus, we have shown the following claim.

Claim B.4.7. *If $\boldsymbol{H}_i$ are uniformly random in all components of $\mathbb{M}_n$, then except with probability at most $2(k-1)/p$, the matrix $\boldsymbol{L}$ is invertible.*

The lemma now follows, since (by induction) the statistical distance is overall at most $2\log_k(n)(k-1)/p$. More formally, each round i is a (probabilistic) function f_i applied to an input distribution D_i. Here

D_i includes all previous $u_{\pm j}$ (for $j = 1, \dots, k-1$), and the current $\boldsymbol{r}$ and $\boldsymbol{H}$.[5] Initially, D_i is uniform. By Claim B.4.7 the round function f_i applied to a uniform distribution has statistical distance of at most $2(k-1)/p$ from a uniform output distribution. Thus, by induction, $f_1(\dots(f_\ell(D_\ell)))$ has statistical distance at most $2\ell(k-1)/p$ from uniform. □

Lemma B.4.8. $\mathsf{LMPA}_{\mathrm{almSnd}}$ *is ε-statistical HVZK for $\varepsilon \in 2m(k-1)\log_k(n)/p$.*

Proof. The base case of the protocol (as part of Step 2), simply relies on Σ_{std}, for which a perfect simulator exists. Thus, we only need to simulate the recursive steps. For any recursive step, it is possible to efficiently compute $[\boldsymbol{u}_0]$ in reverse, given the challenge and all $[\boldsymbol{u}_{\pm\ell}]$ for $\ell = 1, \dots, k-1$. By Lemma B.4.5, all $[\boldsymbol{u}_{\pm\ell}]$ in all rounds are jointly statistically ε-close to uniform for $\varepsilon \le 2m(k-1)\log_k(n)/p$. Thus, if the simulator chooses all $[\boldsymbol{u}_{\pm\ell}]$ in all rounds uniformly, the distribution will be ε-close to an honest execution.

Lastly, we handle Step 1 of the protocol, where $[\boldsymbol{t}^{(i)}]$ for $i = 1, \dots, m$ is sent. Concretely, given $[\boldsymbol{t}]$ and $[\boldsymbol{t}^{(0)}]$ the simulator must find suitable $[\boldsymbol{t}^{(i)}] \in \mathbb{G}^m$ for $i = 1, \dots, m$ such that $[\boldsymbol{t}] = \sum_{i=0}^{m} [\boldsymbol{t}^{(i)}]$. Since $[\boldsymbol{t}^{(i)}]$ is zero, except in the i-th component (for $i \neq 0$), it is uniquely (and efficiently) defined by $[\boldsymbol{t}] - [\boldsymbol{t}^{(0)}]$. This completes the simulation. □

The soundness of Protocol $\mathsf{LMPA}_{\mathrm{almSnd}}$ is weakened by the fact that $\boldsymbol{A}$ is adversarially chosen, and hence finding a non-trivial kernel element of $[\boldsymbol{A} | \boldsymbol{H}'^{(1)} | \dots | \boldsymbol{H}'^{(m)}]$ does not reduce to breaking a hard kernel assumption. This can be avoided by a similar construction as in Protocol $\mathsf{LMPA}_{\mathrm{batch}}$, namely first extending $[\boldsymbol{A}]$ to $[\boldsymbol{A}'] := \left[\begin{smallmatrix} g_0 & \overline{\boldsymbol{g}} \\ 0 & \boldsymbol{A} \end{smallmatrix}\right]$ where $[g_0, \overline{\boldsymbol{g}}] = [\boldsymbol{g}]$, extending $\boldsymbol{w}$ to $\boldsymbol{w}' := (r_{\boldsymbol{w}}, \boldsymbol{w})$ for $r_{\boldsymbol{w}} \xleftarrow{\$} \mathbb{F}_p$, and extending $[\boldsymbol{t}]$ to $[\boldsymbol{t}'] = \left[\begin{smallmatrix} c_{\boldsymbol{w}} \\ \boldsymbol{t} \end{smallmatrix}\right]$ where $[c_{\boldsymbol{w}}] = [g_0]r + [\overline{\boldsymbol{g}}]\boldsymbol{w}$. That is, the prover sends $[c_{\boldsymbol{w}}]$ as its first message. As in Protocol $\mathsf{LMPA}_{\mathrm{batch}}$, we batch down all rows of $[\boldsymbol{A}']$ (except $[\boldsymbol{g}]$) to a single one to reduce communication. Then, with the modified and extended statement $\exists \boldsymbol{w}\colon [\widehat{\boldsymbol{A}}]\boldsymbol{w} = [\widehat{\boldsymbol{t}}]$, prover and verifier engage in $\mathsf{LMPA}_{\mathrm{almSnd}}$. Now, a non-trivial kernel element of $[\widehat{\boldsymbol{A}}, \boldsymbol{H}'^{(1)}, \boldsymbol{H}'^{(2)}]$ implies a non-trivial kernel element of $[\boldsymbol{g}, \boldsymbol{h}]$ (as the first of the two rows is $[\boldsymbol{g}, \boldsymbol{h}, \boldsymbol{0}]$). Since $[\boldsymbol{g}, \boldsymbol{h}]$ is part of the CRS, we can rely on a hard kernel assumption. Thus, this protocol achieves computational soundness. More concretely, the protocol is correct, ε-statistical HVZK with $\varepsilon \le 4(k-1)\log_k(n)/p$, and special sound with short-circuit extraction. We leave the (straightforward) details to the reader.

Lemma B.4.9. *The protocol described above is correct and ε-statistical HVZK with $\varepsilon \le 4(k-1)\log_k(n)/p$. It has $(m+1, 2k-1, \dots, 2k-1, 2)$-special (relaxed) soundness for finding a witness $\boldsymbol{w} \in \mathbb{F}_p^n$ with $[\boldsymbol{A}]\boldsymbol{w} = [\boldsymbol{t}]$, or a non-trivial kernel element of $[\boldsymbol{g}, \boldsymbol{h}]$.*

Moreover, the protocol has $(1, k, \dots, k, 2)$-quick $(m+1, 2k, \dots, 2k, 2)$-short extraction for the same relaxed soundness relation.

Proof. We leave the (straightforward) proofs to the reader. We merely remark that ε is independent of m due to batch-verification (which reduces $[\boldsymbol{A}]$ to 2 rows, i.e. $m = 2$) prior to engaging in the (sub)protocol $\mathsf{LMPA}_{\mathrm{almSnd}}$. □

[5] While we did not explicitly keep track of $\boldsymbol{H}$ before, we must also ensure the after each reduction step, $\widehat{\boldsymbol{H}}$ is again uniformly distributed (in order to apply Claim B.4.7). But that is obvious, by definition of $\boldsymbol{H}$ and $\widehat{\boldsymbol{H}}$.

B.5. Further Notes on Knowledge Soundness and Testing Distributions

This section is a mingle-mangle of further discussion related to knowledge soundness and extraction.

B.5.1. Lower bounds for black-box extraction

In this section, we show that it is unlikely to obtain (significantly) sublinear extraction with *black-box* extraction for many relation of interest. Intuitively, we define a form of hardness which implies "witness incompressibility". A direct consequence is, that an extractor must receive enough "response material" from the prover, or extraction would violate the "incompressibility" assumption. The rest of this section is slightly informal, as the primary goal is merely to point out obstructions for better extractors.

Definition B.5.1. Let $\mathcal{R}$ be an efficient relation and $\mathsf{Sample}(1^\lambda)$ be an efficient algorithm sampling (hard) instances $(\mathbb{x}, \mathbb{w})$ from $\mathcal{R}$. For simplicity, let the size of $(\mathbb{x}, \mathbb{w})$ be the bit-length $|\mathbb{w}|$ of $\mathbb{w}$. Let $\alpha \in \mathbb{N} \to [0,1]$, $t \in \mathbb{N} \to \mathbb{N}$ be efficiently computable functions and consider following game with adversary $(\mathcal{A}, F)$:

1. $(\mathbb{x}, \mathbb{w}) \leftarrow \mathsf{Sample}(1^\lambda)$
2. Compute the hint $h = F(1^\lambda, \mathbb{x}, \mathbb{w}; r)$, where r is the random tape of F.
3. $\mathcal{A}(1^\lambda, \mathbb{x}, h, r)$ must output some $\mathbb{w}'$. If $(\mathbb{x}, \mathbb{w}') \in \mathcal{R}$, return 1 ($\mathcal{A}$ wins) else 0 ($\mathcal{A}$ loses).

The advantage of the pair $(\mathcal{A}, F)$ is defined as the probability that $\mathcal{A}$ wins the game. We say $\mathcal{R}$ is (α, t)-hard w.r.t. Sample, if for every pair of t-time algorithms $(\mathcal{A}, F)$ playing the above game, the advantage of $\mathcal{A}$ is negligible.

Corollary B.5.2. *Let* Ext *be an expected polynomial time black-box extractor with negligible knowledge error for some efficient proof system* (GenCRS, P, V) *for relation $\mathcal{R}$ which is (α, t)-hard for superpolynomial t (or for every polynomial t). Let $s = s(\lambda)$ bound the size of the responses* Ext *received from its black-box access to the* honest *prover* P. *Then,* $3s(\lambda) > \alpha(|\mathbb{w}|)|\mathbb{w}|$

Note that we bound the required number of transcripts for *honest* provers. Clearly, for malicious provers, the situation only gets worse.

Proof sketch. Let $\varepsilon(\lambda)$ be the success probability of Ext in the knowledge soundness experiment with the honest prover. Since we consider honest provers and negligible knowledge error, $\varepsilon(\lambda)$ is overwhelming, so at least $1/2$ for large enough λ. Suppose also that Ext has expected time bounded by a polynomial $t_{\mathsf{Ext}}(\lambda)$. Since $\mathcal{R}$ is (α, t)-hard w.r.t. Sample for every polynomial t, and since the proof system is efficient, our idea is to run the extraction experiment within F. The output of F consists of all messages which Ext would have received from its prover oracle. Since F's randomness is always passed to $\mathcal{A}$, the view of Ext is available and $\mathcal{A}$ can re-run the experiment, in particular re-run Ext, to compute $\mathbb{w}'$. Whenever Ext succeeds, so does $\mathcal{A}$. Since F is strictly t-time, but Ext is an expected time algorithm, we must truncate the execution of Ext in F to make this idea work. By Markov's inequality, truncating the execution of Ext after $2t_{\mathsf{Ext}}/\varepsilon$ steps will only lead to a timeout with probability $\frac{\varepsilon}{2}$. Since $\varepsilon(\lambda) \geq 1/2$ for large enough λ, we can let F truncate Ext after $t = 4t_{\mathsf{Ext}}$ and get a success probability of $\mathcal{A}$ which is at least $\frac{\varepsilon}{2}$ infinitely often. Lastly, since the responses which Ext received from its black-box access to P (in the simulation of F) have a length of $s(\lambda)$ bits, they can be encoded in the output of F in at most $3s(\lambda)$ bits.

All in all, we have constructed a t-time adversary $(\mathcal{A}, F)$ from $\mathcal{A}$ with non-negligible success probability. Since $\mathcal{R}$ is (α, t)-hard w.r.t. Sample for any polynomial t, this implies (by contradiction) that $3s(\lambda) > \alpha(|\mathbb{w}|)|\mathbb{w}|$. □

As a consequence of Corollary B.5.2, we see that if a there exists a system of quadratic equations with (α, poly)-hardness for some constant α and all polynomials poly, then to extract a witness of dimension n, hence size $m = \mathcal{O}(n\lambda)$ (due to $\mathbb{F}_p^n$ with $\log(p) \in \mathcal{O}(\lambda)$), the total response size must be $\Omega(\alpha m)$. In an extraction tree of depth d, every leaf in the tree corresponds to at most d responses. In fact, in general, much fewer since most paths in the tree share some responses. If every response has maximal size $s \in \mathcal{O}(\lambda)$, the number of leaves must be $\Omega(\frac{\alpha m}{ds}) = \Omega(\frac{\alpha n}{d})$.

To conclude, we see that under certain (very strong) hardness assumptions, extractors for succinct proofs must query their oracle often to be able to extract. While such assumptions may not hold for actual problems of interest, they present a significant obstruction for "generic" means of extraction, which do not explicitly exploit the "compressibility" of the witness. To circumvent these obstructions, very strong cryptographic hardness assumptions or model assumptions seem necessary, e.g. extractable hash functions or random oracles, seem to be necessary. With such powerful cryptographic assumptions it is well-known how to achieve succinct proofs with straight-line extractability.[6]

B.5.2. Properties of Testing Distributions

Remark B.5.3. Let χ_m be a testing distribution. Then the probability that $x_i = x_j$ for $\boldsymbol{x} \xleftarrow{\$} \chi_m$ is smaller than $\delta_{\mathsf{snd}}(\chi_m)$. This holds since the vectors with $x_i = x_j$ are the set $B = \ker(\boldsymbol{z}^\top)$ for $\boldsymbol{z} = \boldsymbol{e}_i - \boldsymbol{e}_j \neq \boldsymbol{0}$. And $\Pr_{\boldsymbol{x} \xleftarrow{\$} \chi}[\boldsymbol{x}^\top \boldsymbol{z}] \leq \delta_{\mathsf{snd}}(\chi)$ holds by definition.

Remark B.5.4. The above argument in Remark B.5.3 generalises to other relations/properties of vectors which affect invertibility. Thus, a testing distribution must be "well-spread" over a vector space to achieve high information-theoretic soundness.

Remark B.5.5. As another measure for the soundness error of a testing distribution χ_m, one may look at the probability $\Pr_{\boldsymbol{X} \xleftarrow{\$} \chi_m^m}[\det(\boldsymbol{X}) \neq 0]$, i.e. the probability that m challenges are linearly independent. This measure, while clearly related to our definition of information-theoretic soundness error, behaves quite differently. For example, consider a testing distribution which picks uniformly from $\{0, 1\}^m$. A well-known conjecture for random *binary* $n \times n$ matrices over the *reals*, states that only a $(1 + o(1))n^2 2^{-n}$ fraction is singular. However, the probability that $x_i = x_j$ is $\frac{1}{4}$ in this case, and hence by Remark B.5.3 the information-theoretic soundness error is at least $1/4$. While in our case, the matrices are *not* over the reals, but modulo p, it is very plausible that only a small portion, perhaps $\mathcal{O}(1/p)$, of random binary matrices fail to be invertible over $\mathbb{F}_p$ but are invertible over $\mathbb{R}$ — indeed, this failure happens if and only if the determinant $\det(\boldsymbol{X}) \in p\mathbb{Z}$.

Remark B.5.6. Instead of analyzing testing distributions over $\mathbb{F}_p^m$, one can do so over the projective space $(\mathbb{F}_p^m \setminus \{\boldsymbol{0}\})/\mathbb{F}_p^\times$, i.e. one can identify all "test vectors" which are non-zero scalar multiples of another and exclude $\boldsymbol{0}$. The main reason this works is, that linear independence can be checked on these equivalence classes, and the information-theoretic soundness error as well. By fixing one component to 1 (as e.g. in monomimial testing distributions), one already ensures no scalar multiples exist.

[6] While straight-line extractability does not imply tightness in the usual sense, e.g. "extracting" a hash function may be very expensive, it is a first step towards the goal. In the ROM, extractability is free.

B.5.3. Tensor-Based Testing Distributions

Recall that, in a sense, construction of tensor-based testing distributions corresponds to multiple challenges, or the unrolling of the recursive steps in our proof systems. The following lemma tightly characterizes their information-theoretic soundness error.

Lemma B.5.7. *Let $\chi = \chi_1 \otimes \ldots \otimes \chi_\ell$ be the tensor product of ℓ testing distributions χ_i on $\mathbb{F}_p^{k_i}$ with $\delta_{\mathsf{snd}}(\chi_i)$. Then χ has*

$$\delta_{\mathsf{snd}}(\chi) \leq 1 - \prod_{i=1}^{\ell}(1 - \delta_{\mathsf{snd}}(\chi_i)) \leq \sum_{i=1}^{\ell} \delta_{\mathsf{snd}}(\chi_i).$$

Proof. By induction, it suffices to consider $\ell = 2$. Let $\delta_i := \delta_{\mathsf{snd}}(\chi_i)$. Recall that

$$\delta_{\mathsf{snd}}(\chi) = \max_{\boldsymbol{z}} \Pr_{\boldsymbol{x} \xleftarrow{\$} \chi}[\boldsymbol{x}^\top \boldsymbol{z} = 0] = \max_{H} \Pr_{\boldsymbol{x} \xleftarrow{\$} \chi}[\boldsymbol{x} \in H]$$

where H ranges over all hyperplanes. Recall that for every hyperplane $H \leq \mathbb{F}_p^m$, there is a (non-zero) linear maps $\phi \colon \mathbb{F}_p^m \to \mathbb{F}_p^m$ with $H = \ker(\phi)$. That is, $\phi(\boldsymbol{x}) = 0 \iff \boldsymbol{x} \in H$. Let $\varphi \colon \mathbb{F}_p^{k_1} \otimes \mathbb{F}_p^{k_2} \to \mathbb{F}_p$ be such a linear map and $H = \ker(\varphi)$. Recall that any element $\boldsymbol{z}$ in $\mathrm{supp}(\chi)$ is an *elementary tensor* $\boldsymbol{x} \otimes \boldsymbol{y}$ by definition of $\chi = \chi_1 \otimes \chi_2$.

Considering the induced linear map $\varphi(\,\cdot\, \otimes \boldsymbol{y}) \colon \mathbb{F}_p^{k_1} \to \mathbb{F}_p$, we find that, for any choice of $\boldsymbol{y}$,

$$\Pr_{\boldsymbol{x} \leftarrow \chi_1}[\varphi(\boldsymbol{x} \otimes \boldsymbol{y}) = 0] \leq \delta_1$$

by definition of $\delta_{\mathsf{snd}}(\chi_1)$, except if $\varphi(\_ \otimes \boldsymbol{y}) = \boldsymbol{0}$ as a map. But $\varphi(\_ \otimes \boldsymbol{y}) = \boldsymbol{0}$ implies

$$\boldsymbol{y} \in K := \{\boldsymbol{b} \mid \varphi(\_ \otimes \boldsymbol{b}) = \boldsymbol{0}\} \leq \mathbb{F}_p^{k_2}.$$

Observe that K is a subspace of dimension at most $(k_2 - 1)$, (else $\varphi = 0$, a contradiction). Thus, we get

$$\Pr_{\boldsymbol{y} \leftarrow \chi_2}[\varphi(\_ \otimes \boldsymbol{y}) = \boldsymbol{0}] = \Pr_{\boldsymbol{y} \leftarrow \chi_2}[\boldsymbol{y} \in K] \leq \delta_2.$$

Then we from $\boldsymbol{z} = \boldsymbol{x} \otimes \boldsymbol{y}$ that

$$\begin{aligned}
\Pr_{\boldsymbol{z} \leftarrow \chi}[\boldsymbol{z} \in V] &= \Pr_{\boldsymbol{x} \xleftarrow{\$} \chi_1, \boldsymbol{y} \xleftarrow{\$} \chi_2}[\varphi(\boldsymbol{x} \otimes \boldsymbol{y}) = 0] \\
&\leq (1 - \Pr_{\boldsymbol{y} \xleftarrow{\$} \chi_2}[\boldsymbol{y} \notin K]) \max_{\boldsymbol{y} \notin K} \Pr_{\boldsymbol{x} \xleftarrow{\$} \chi_1}[\varphi(\boldsymbol{x} \otimes \boldsymbol{y})] + \Pr_{\boldsymbol{y}}[\boldsymbol{y} \in K] \\
&\leq (1 - \delta_2)\delta_1 + \delta_2 \\
&= 1 - (1 - \delta_1)(1 - \delta_2) \\
&\leq \delta_1 + \delta_2.
\end{aligned}$$

The claim follows for $\ell = 2$ and by straightforward induction for general ℓ. □

Our recursive arguments actually have a tensor structure, namely they reduce $\mathbb{F}_p^n = (\mathbb{F}_p^k)^\ell$ to $(\mathbb{F}_p^k)^{\ell-1}$ in one step, i.e. they apply a linear map to one of the factors of the tensor product. It is not hard to see that in Section 4.3.6, Protocol 4.3.18, one applies $\boldsymbol{x}_1 \otimes \ldots \otimes \boldsymbol{x}_\ell$ to $[\boldsymbol{A}]$ and $\boldsymbol{y}_1 \otimes \ldots \otimes \boldsymbol{y}_\ell$ to $\boldsymbol{w}$ when all batching steps are taken together.

B.6. Further Remarks on our Implementation

B.6.1. Arithmetic Circuits

We use $\mathsf{QESA}_{\mathsf{ZK}}$ to prove arithmetic circuits. In contrast to existing techniques, $\mathsf{QESA}_{\mathsf{ZK}}$ is not restricted to R1CS circuits, but can also handle quadratic equations. Hence we include a preprocessing step in Python, which transforms arithmetic circuits generated by the Pinocchio compiler [PGHR13] or jsnark[7] into quadratic equations.

Preprocessing. We preprocess the arithmetic circuit in order to better make use of "quadratic equation gates" (quad gates in the following). To this end, we perform a series of transformations, which in the end yield an equivalent circuit comprised almost entirely of quad gates.

The transformations follow a few simple observations. Some gates can be represented directly by (quadratic) constraints. For example, $\mathsf{xor}(X, Y)$ can be represented as $(1 - X)Y + X(1 - Y) = 0$. We refer to these as isolated gates in the following. Other gates, such as pack with $\mathsf{pack}(x_1, \ldots, x_k) = \sum_1^k x_i 2^i = x_0 + 2(x_1 + 2(\ldots + 2x_k \ldots))$, can be decomposed into a series of arithmetic gates, hence we coin them decomposable gates. The remaining basic gates, i.e., add, sub, const-mul, and const-mul-neg, can be merged if they precede a mul gate, resulting in a quad gate computing $\sum_{i,j} w_i \Gamma_{i,j} w_j = w_k$. Such a quad gate $\mathfrak{g}$ can be represented by $\mathbf{\Gamma}_\mathfrak{g} = \sum_i \boldsymbol{a}_{\mathfrak{g},i} \boldsymbol{b}_{\mathfrak{g},i}^\top - \boldsymbol{e}_\mathfrak{G} \boldsymbol{e}_\mathfrak{g}^\top \in \mathbb{F}_p^{n \times n}$, where $\boldsymbol{a}_{\mathfrak{g},i}$, $\boldsymbol{b}_{\mathfrak{g},i}$ are constants describing the gate. We find that $\boldsymbol{w}^\top \mathbf{\Gamma}_\mathfrak{g} \boldsymbol{w} = 0$ iff $\mathfrak{g}$ is satisfied by the wire assignment $\boldsymbol{w}$.

Based on these observations, our preprocessing applies the following steps: First, decomposable gates are replaced with other gates depending on their functionality.

Then, each wire w that is either a global output wire or an input wire of an isolated gate, is prepended with a new mul gate where one input is w and the other is the constant-1 wire. Naturally, this is only applied if w is not already the output of a mul gate. The insertion allows for later aggregation of preceding logic into a single quad gate.

Now, all remaining basic gates are merged into quad gates of the form $\sum_{i,j} a_i w_i \Gamma_{i,j} b_j w_j = w_k$. This aggressive optimisation may result in several gates with constant $w_k = 0$. Therefore, constant zeros are propagated through the circuit, eliminating affected gates and wires. Finally the circuit is stripped of floating gates where no output is connected any more and for each remaining gate the corresponding $\mathbf{\Gamma}_i$ is extracted.

Results. We evaluate $\mathsf{QESA}_{\mathsf{ZK}}$ using the same 512-bit SHA256 circuit without padding as in [BBB+18]. The preprocessed circuit consists of 25657 wires, i.e. , $\boldsymbol{w} \in \mathbb{F}_p^{25657}$ and 25840 matrices $\mathbf{\Gamma}_i \in \mathbb{F}_p^{25657 \times 25657}$. If the $\mathbf{\Gamma}_i$ would have been stored without the sparse matrix optimisation, this would require the implementation to hold $25840 \cdot 25657^2 > 2^{43}$ $\mathbb{F}_p$ elements in memory just for the matrices. The sparse representation reduces this to 197465 $\mathbb{F}_p$ elements. Since $\mathsf{QESA}_{\mathsf{ZK}}$ expects n to be a power of two, we set $n = 2^{15} = 32768$ and the witness is zero-extended accordingly. As a result, the implementation took 84.2s for P and 38.1s for V on average.

[7] See: `https://github.com/akosba/jsnark`

Parameters	Bulletproofs		Bulletproofs with $\mathsf{IPA}_{\mathrm{noZK}}$	
	P	V	P	V
60 bit	0.26	0.17	0.23	0.11
60 bit × 2	0.47	0.29	0.42	0.21
60 bit × 32	7.4	4.5	6.3	3.7
60 bit × 128	28.9	17.9	26.6	14.2
60 bit × 512	116	78.7	105	55.5
124 bit	0.46	0.29	0.41	0.22
124 bit × 32	14.9	9.2	13.6	7.0
124 bit × 128	59.7	36.8	54.1	29.7
124 bit × 512	238	147	219	117
252 bit	0.95	0.59	0.79	0.46
252 bit × 32	30.2	18.6	26.1	14.3
252 bit × 128	121	74.3	105	58.4
252 bit × 512	484	297	426	227

Table B.1.: Comparison of non-optimised prover runtime in seconds of aggregate range proofs from [BBB+18] with the original IPA and with $\mathsf{IPA}_{\mathrm{noZK}}$. Verification times are only included for completeness, since powerful optimziations equalize efficiency, see Section 4.5 for details.

B.6.2. Bulletproofs with $\mathsf{IPA}_{\mathrm{noZK}}$

One of our main contributions is the improvement of the original IPA from [BBB+18]. In order to practically evaluate the impact of said improvements, we benchmarked Bulletproofs aggregate range proofs with the same parameters as in Table 4.3, but this time used $\mathsf{IPA}_{\mathrm{noZK}}$ instead. Table B.1 shows the results.

B.7. Overview of Protocols

In the following, we give an overview of the protocols for with several choices fixed. In particular, we fix $k = 2$. Otherwise, the respective setting is as in the definition of the protocols. Let $\mathcal{S} \subseteq \mathbb{F}_p^\times$. Note that $\mathcal{S}$ does *not contain* zero. For simplicity, we use the testing distribution $\chi^{(\beta\neq 0)}$, which draws $\alpha \xleftarrow{\$} \mathcal{S}$ and returns $(\alpha, 1)$. (Indeed, $\chi^{(\beta\neq 0)} = \chi^{(\beta)}$ since $0 \notin \mathcal{S}$.) Moreover, we write $\alpha \xleftarrow{\$} \chi^{(\beta\neq 0)}$ instead. For other testing distributions χ_n, we consider $\boldsymbol{x} \xleftarrow{\$} \{1\} \times \mathcal{S}^{n-1}$, that is $x_1 = 1$ always and the other components are random (small) exponents in $\mathcal{S}$. These choices are compatible with the restrictions posed in some protocols. For $\widetilde{\chi}_{2k-1}$ we use an explicit choice $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$, namely $(1, \beta) = \boldsymbol{x} \xleftarrow{\$} \chi^{(\beta\neq 0)}$, $\boldsymbol{y} = (\beta, 1)$ and $\boldsymbol{z} = (1, \beta, \beta^2)$.

In our presentation, we use following conventions:

- Inputs, which *must* be known to *both* parties are *common* inputs.
- Inputs, which a party (generally the prover) can derive from other inputs, are removed from common inputs.

For example, the target value t in $\mathsf{IPA}_{\mathrm{almZK}}$ is not a treated as a common input, since P can recompute $t = \langle \boldsymbol{w}', \boldsymbol{w}'' \rangle$ via the witness. This makes data flow (and some optimisations) more explicit, e.g. Remark 4.4.8.

$\mathsf{IPA}_{\mathrm{noZK}}$(Protocol 4.4.1)		
	Common Input: $crs = ([\boldsymbol{g}', \boldsymbol{g}'', Q])$	
Prover P		Verifier V
Input: $\boldsymbol{w}', \boldsymbol{w}''$		Input: $[c]$, t
		$\alpha \overset{\$}{\leftarrow} \chi^{(\beta \neq 0)}$
	$\xleftarrow{\alpha}$	
$[Q] := \alpha^{-1}[Q]$		$[Q] := \alpha^{-1}[Q]$ $[c] := [c] - (\alpha - 1)t[Q]$
	Recursive step. Suppose $n > 1$	
split $\boldsymbol{w}'$ in halves $\boldsymbol{w}'_1, \boldsymbol{w}'_2$ split $\boldsymbol{w}'', \boldsymbol{g}', \boldsymbol{g}''$ analogously $[\boldsymbol{u}'_{-1}] := [\boldsymbol{g}'_2]\boldsymbol{w}'_1$, $[\boldsymbol{u}'_{+1}] := [\boldsymbol{g}'_1]\boldsymbol{w}'_2$ compute $[\boldsymbol{u}''_{\pm 1}]$ analogously $v_{-1} := \langle \boldsymbol{w}'_2, \boldsymbol{w}''_1 \rangle$ $v_{+1} := \langle \boldsymbol{w}'_1, \boldsymbol{w}''_2 \rangle$ $[\boldsymbol{u}_{-1}] := [\boldsymbol{u}'_{-1}] + [\boldsymbol{u}''_{+1}] + v_{+1}[Q]$ $[\boldsymbol{u}_{+1}] := [\boldsymbol{u}'_{+1}] + [\boldsymbol{u}''_{-1}] + v_{-1}[Q]$		
	$\xrightarrow{[\boldsymbol{u}_{-1}],[\boldsymbol{u}_{+1}]}$	
		$\xi \overset{\$}{\leftarrow} \chi^{(\beta \neq 0)}$
	$\xleftarrow{\xi}$	
$[\boldsymbol{g}'] := [\boldsymbol{g}'_1] + \xi[\boldsymbol{g}'_2]$ $[\boldsymbol{g}''] := \xi[\boldsymbol{g}''_1] + [\boldsymbol{g}''_2]$ $\boldsymbol{w}' := \xi\boldsymbol{w}'_1 + \boldsymbol{w}'_2$ $\boldsymbol{w}'' := \boldsymbol{w}''_1 + \xi\boldsymbol{w}''_2$ $n := n/2$		$[\boldsymbol{g}'] := [\boldsymbol{g}'_1] + \xi[\boldsymbol{g}'_2]$ $[\boldsymbol{g}''] := \xi[\boldsymbol{g}''_1] + [\boldsymbol{g}''_2]$ $[c] := \xi^2[\boldsymbol{u}_{-1}] + \xi[c] + [\boldsymbol{u}_{+1}]$ $n := n/2$
	Start next recursion iteration.	
	Base case. Suppose $n = 1$	
	$\xrightarrow{\boldsymbol{w}', \boldsymbol{w}''}$	
		return true iff: $[c] \overset{?}{=} [\boldsymbol{g}']\boldsymbol{w}' + [\boldsymbol{g}'']\boldsymbol{w}'' + t[Q]$ where $t := \langle \boldsymbol{w}', \boldsymbol{w}'' \rangle$

B.8. A Short Note on R1CS and QE

The work [AC20] uses a simple linearization technique to replace the inner product argument (IPA), by its basic building block, a linear map preimage argument (LMPA). The improvement of [AC20] is mostly conceptual and does not offer efficiency improvements (computation- or communication-wise). The encoding of [AC20] is based on multiplication-triples, and it does not seem to immediately generalize to systems of (general) quadratic equations (QE), and appears limited to rank-1 constraint systems (R1CS).

In this section,

- we point out a simple way to efficiently reduce QE to R1CS (almost for free);

$\mathsf{IPA}_{\mathrm{almZK}}$ (Protocol 4.4.3)

Common Input: $crs = ([\boldsymbol{g}', \boldsymbol{g}'', Q])$

Prover P		Verifier V
Input: $\boldsymbol{w}', \boldsymbol{w}''$		Input: $[c_{\boldsymbol{w}}]$, t
$\boldsymbol{r}' \xleftarrow{\$} \ker(\boldsymbol{w}''^{\top}) \cap \mathbb{M}_n^+$		
$\boldsymbol{r}'' \xleftarrow{\$} \left\{ \boldsymbol{v} \in \mathbb{M}_n^+ \middle\vert \begin{array}{l} \langle \boldsymbol{r}', \boldsymbol{r}'' \rangle = 0 \\ \wedge \langle \boldsymbol{w}', \boldsymbol{r}'' \rangle = -\langle \boldsymbol{r}', \boldsymbol{w}'' \rangle \end{array} \right\}$		
$[c_{\boldsymbol{r}}] := [\boldsymbol{g}']\boldsymbol{r}' + [\boldsymbol{g}'']\boldsymbol{r}''$		
	$\xrightarrow{[c_{\boldsymbol{r}}]}$	
		$\beta \xleftarrow{\$} \chi^{(\beta)}$
	$\xleftarrow{\beta}$	
		$t := \beta^2 t$
$\boldsymbol{w}' := \beta \boldsymbol{w}' + \boldsymbol{r}'$		$[c] = [c_{\boldsymbol{r}}] + \beta[c_{\boldsymbol{w}}] + t[Q]$
$\boldsymbol{w}'' := \beta \boldsymbol{w}'' + \boldsymbol{r}''$		

Engage $\mathsf{IPA}_{\mathrm{noZK}}(crs, \mathsf{P}(\boldsymbol{w}', \boldsymbol{w}''), \mathsf{V}([c], t))$

$\mathsf{QESA}_{\mathrm{Inner}}$ (part of Protocol 4.4.7)

Common Input: $crs = ([\boldsymbol{g}', \boldsymbol{g}'', Q]), \{\boldsymbol{\Gamma}_i\}$

Prover P		Verifier V
Input: $\boldsymbol{w}, \boldsymbol{r}'$		Input: $[c'_{\boldsymbol{w}}]$
$\boldsymbol{w}' := \binom{\boldsymbol{w}}{\boldsymbol{r}'}$		$\boldsymbol{x} \xleftarrow{\$} \chi_N$
	$\xleftarrow{\boldsymbol{x}}$	
$\boldsymbol{\Gamma} := \sum_{i=1}^{N} x_i \boldsymbol{\Gamma}_i$		$\boldsymbol{\Gamma} := \sum_{i=1}^{N} x_i \boldsymbol{\Gamma}_i$
$\beta := x_2$		$\beta := x_2$
$[g'_1] := \beta^{-1}[g'_1]$		$[g'_1] := \beta^{-1}[g'_1]$
$\boldsymbol{w}'' := \binom{\boldsymbol{\Gamma}\boldsymbol{w}}{\boldsymbol{R}\boldsymbol{r}'}$		$[c'_{\boldsymbol{w}}] := [c'_{\boldsymbol{w}}] - (\beta - 1)[g'_1]$
$[c''_{\boldsymbol{w}}] := [\boldsymbol{g}'']\boldsymbol{w}''$		
	$\xrightarrow{[c''_{\boldsymbol{w}}]}$	
		$(1, \boldsymbol{s}, \boldsymbol{b}) \xleftarrow{\$} \chi_n$, $\boldsymbol{s}' := \binom{\boldsymbol{s}}{\boldsymbol{b}}$
	$\xleftarrow{\boldsymbol{s}'}$	
		$t := -\langle \boldsymbol{s}, \boldsymbol{\Gamma}^{\top} \boldsymbol{s} \rangle$
$\boldsymbol{w}' := \boldsymbol{w}' - \boldsymbol{s}'$		$[c_{\boldsymbol{w}}] := [c'_{\boldsymbol{w}}] - [\boldsymbol{g}']\boldsymbol{s}' + [c''_{\boldsymbol{w}}] + [\boldsymbol{g}'']\boldsymbol{\Gamma}'^{\top}\boldsymbol{s}'$
$\boldsymbol{w}'' := \boldsymbol{w}'' + \boldsymbol{\Gamma}'^{\top}\boldsymbol{s}'$		

Engage $\mathsf{IPA}_{\mathrm{almZK}}(crs, \mathsf{P}(\boldsymbol{w}', \boldsymbol{w}''), \mathsf{V}([c_{\boldsymbol{w}}], t))$

- we show a direct encoding of for QE which does not go through multiplication-triples/Lagrange basis (but still multiplication of univariate polynomials of high degree, unlike IPA-based encodings of R1CS or QE).

Recall that, while R1CS and QE are closely related (and both NP-complete), there is a difference in expressivity. For example, when an inner product (e.g. the 2-norm) between n-dimensional vector must be computed, this requires $n - 1$ auxiliary variables (and constraints) in R1CS, but no auxiliary variables

$\mathrm{QESA}_{\mathrm{ZK}}$ (Protocol 4.4.7)

Common Input: $crs = ([\boldsymbol{g}', \boldsymbol{g}'', Q]), \{\boldsymbol{\Gamma}_i\}$

Prover P		Verifier V
Input: $\boldsymbol{w}$		Input: $\emptyset$
$\boldsymbol{r}' \xleftarrow{\$} \mathbb{F}_p^2$		
$[c'_{\boldsymbol{w}}] := [\boldsymbol{g}']\binom{\boldsymbol{w}}{\boldsymbol{r}'}$		
	$\xrightarrow{[c'_{\boldsymbol{w}}]}$	

Engage $\mathrm{QESA}_{\mathrm{Inner}}((crs, \{\boldsymbol{\Gamma}_i\}), \mathsf{P}(\boldsymbol{w}, \boldsymbol{r}'), \mathsf{V}([c'_{\boldsymbol{w}}]))$

$\mathrm{QESA}_{\mathrm{Copy}}$(Protocol 4.4.17)

Common Input: $crs = ([\boldsymbol{g}', \boldsymbol{g}'', Q]), \{\boldsymbol{\Gamma}_i\}, \{\widetilde{ck}^{(i)}\}, \{[\widetilde{c}^{(i)}]\}$

Prover P		Verifier V
Input: $\boldsymbol{w}, \{\boldsymbol{v}^{(i)}\}$		Input: $\emptyset$
$\boldsymbol{r}' \xleftarrow{\$} \mathbb{F}_p^2$		
$\boldsymbol{w}' := \binom{\boldsymbol{w}}{\boldsymbol{r}'}$		
$[c'_{\boldsymbol{w}}] := [\boldsymbol{g}']\boldsymbol{w}'$		
	$\xrightarrow{[c'_{\boldsymbol{w}}]}$	
		$\boldsymbol{\alpha} \xleftarrow{\$} \chi_{M+1}$ with $\alpha_0 = 1$
	$\xleftarrow{\boldsymbol{\alpha}}$	
$[c'_{\boldsymbol{w}}] := \alpha_0[c'_{\boldsymbol{w}}] + \sum_{i=1}^{M} \alpha_i[\widetilde{c}^{(i)}]$		$[c'_{\boldsymbol{w}}] := \alpha_0[c'_{\boldsymbol{w}}] + \sum_{i=1}^{M} \alpha_i[\widetilde{c}^{(i)}]$
$\{\boldsymbol{\Gamma}_i\}_i := \{\boldsymbol{\Gamma}_i\}_i \cup \{\boldsymbol{\Gamma}^{(k)}_{\mathrm{copy}}$ for $k \in \mathcal{G}\}$		$\{\boldsymbol{\Gamma}_i\}_i := \{\boldsymbol{\Gamma}_i\}_i \cup \{\boldsymbol{\Gamma}^{(k)}_{\mathrm{copy}}$ for $k \in \mathcal{G}\}$
$\boldsymbol{w}' := \alpha_0\boldsymbol{w}' + \sum_{i=1}^{M} \alpha_i\boldsymbol{v}^{(i)}$		
decompose $(\boldsymbol{w}, \boldsymbol{r}') := \boldsymbol{w}'$		

Engage $\mathrm{QESA}_{\mathrm{Inner}}((crs, \{\boldsymbol{\Gamma}_i\}), \mathsf{P}(\boldsymbol{w}, \boldsymbol{r}'), \mathsf{V}([c'_{\boldsymbol{w}}]))$

(and a single constraint) in QEs. As such, it can be beneficial to express equations in terms of QE, even if the underlying proofs systems reduces QE to R1CS (as per our first suggestion).

B.8.1. Preliminaries

Let p be an (odd) (prime) number. Let $\mathbb{Z}_p = \mathbb{Z}/p\mathbb{Z}$ be the integers modulo p. We write R for some ring. For simplicity, we assume $R = \mathbb{Z}_p$, but it is sufficient if R is well-behaved w.r.t. polynomials, Lagrange interpolation and zero-testing with Schwartz–Zippel.

B.8.2. Comparing R1CS and QE

We will look at the question of how to linearize arbitrary quadratic equations (QE), and how systems of QEs behave relative to rank 1 constraint systems (R1CS) [BCG+13]. Recall that a R1CS-type constraint is of the form $(\sum_{i=1}^{n} a_i w_i) \cdot (\sum_{i=1}^{n} b_i w_i) = \sum_{i=1}^{n} c_i w_i$, where $a_i, b_i, c_i \in R$ are constants and $\boldsymbol{w} \in R^n$ is the witness. Hence R1CS only allows "one multiplication per equation". General quadratic equations are of

$\mathsf{LMPA}_{\mathrm{noZK}}$(Protocol 4.3.18)

Common Input: $[\boldsymbol{A}]$

Prover P Input: $\boldsymbol{w}$		Verifier V Input: $[\boldsymbol{t}]$
	Recursive step. Suppose $n > 1$	
$[\boldsymbol{u}_{-1}] := [\boldsymbol{A}_1]\boldsymbol{w}_2$ $[\boldsymbol{u}_{+1}] := [\boldsymbol{A}_2]\boldsymbol{w}_1$		
	$\xrightarrow{[\boldsymbol{u}_{-1}],[\boldsymbol{u}_{+1}]}$	
		$\xi \xleftarrow{\$} \chi^{(\beta\neq0)}$
	$\xleftarrow{\xi}$	
$[\boldsymbol{A}] := [\boldsymbol{A}_1] + \xi[\boldsymbol{A}_2]$ $\boldsymbol{w} := \xi\boldsymbol{w}_1 + \boldsymbol{w}_2$ $n := n/2$		$[\boldsymbol{A}] := [\boldsymbol{A}_1] + \xi[\boldsymbol{A}_2]$ $[\boldsymbol{t}] := [\boldsymbol{u}_{-1}] + \xi[\boldsymbol{t}] + \xi^2[\boldsymbol{u}_{+1}]$ $n := n/2$
	Start next recursion iteration.	
	Base case. Suppose $n = 1$	
	$\xrightarrow{\boldsymbol{w}}$	
		return true iff $[\boldsymbol{A}]\boldsymbol{w} \overset{?}{=} [\boldsymbol{t}]$

$\mathsf{LMPA}_{\mathrm{simpleZK}}$

Common Input: $[\boldsymbol{A}]$

Prover P Input: $\boldsymbol{w}$		Verifier V Input: $[\boldsymbol{t}]$
$\boldsymbol{r} \xleftarrow{\$} \mathbb{F}_p^n$ $[\boldsymbol{a}] := [\boldsymbol{A}]\boldsymbol{r}$		
	$\xrightarrow{[\boldsymbol{a}]}$	
		$\beta \xleftarrow{\$} \chi^{(\beta\neq0)}$
	$\xleftarrow{\beta}$	
	Engage $\mathsf{LMPA}_{\mathrm{noZK}}([\boldsymbol{A}], \mathsf{P}(\beta\boldsymbol{w} + \boldsymbol{r}), \mathsf{V}(\beta[\boldsymbol{t}] + [\boldsymbol{a}]))$	

the form $\boldsymbol{w}^\top\boldsymbol{\Gamma}\boldsymbol{w} = 0$ for $\boldsymbol{w} \in R^n$, $\boldsymbol{\Gamma} \in R^{n\times n}$. For our constructions, we also allow $\boldsymbol{x}^\top\boldsymbol{\Gamma}\boldsymbol{y}$ in the following, and similarly "relax" R1CS.[8]

Clearly, any QE can be encoded as a *set* of R1CS-type equations. But this introduces an overhead, e.g. an inner product of size n is *one* QE, but $n - 1$ R1CS equations are required (at least naively).

In the rest of this section, we briefly discuss the following:

1. A simple and relatively "black-box" reduction from QE to R1CS. This comes with a small overhead for *suitable* argument systems (e.g., typical commit-and-prove argument systems). Namely, it

[8] Since $x_i = y_i$ is expressible as R1CS resp. QE, this does not affect the expressivity much, yet the number of variables doubles.

introduces auxiliary variables (but at most doubles the number of variables). It is explained in Remark B.8.1 below.

2. A (polynomial encoding) linearization strategy which allows to prove QEs instead of R1CS directly, using the low-level primitives. For this, we use an inner product encoding, which allows to switching out R1CS with QEs basically for free. This is discussed in Appendix B.8.3.
3. A QE encoding which is applicable only in very special settings. It exploits strong symmetries to reduce the amount of auxiliary variables by 50%. It is discussed in Appendix B.8.4.
4. In Remark B.8.2 we briefly highlight the effect of the languages considered, and how this is perhaps the biggest factor in the comparison of R1CS and QE, and how it affects the variants and tricks we consider. It is discussed more in Appendix B.8.5.

In the above and the following, we have in mind an application with commit-and-prove systems (or an ideal linear commitment (ILC) setting). In that setting, one can verify any number of R1CS-type equations (or QE-type equations) roughly at the price of one (and with small soundness loss) by using probabilistic batch-verification techniques.

Remark B.8.1 (Simple efficient "black-box" reduction to R1CS). To transform QE to R1CS naively, one introduces a new variable for every product. This has a quadratic overhead. By writing the quadratic equation $\boldsymbol{w}^\top \boldsymbol{\Gamma} \boldsymbol{w} = t$ as $\boldsymbol{w}^\top(\boldsymbol{\Gamma}\boldsymbol{w}) = t$ for $\boldsymbol{\Gamma} \in R^{m \times n}$, it is becomes evident that $\min(m,n)$ auxiliary variables and R1CS equations suffice. Namely, $w_i \cdot (\boldsymbol{\Gamma}\boldsymbol{w})_i = z_i$ and $\sum_{i=1}^n z_i = t$ is an R1CS system for the quadratic equation, where the z_i are auxiliary variables.[9] In the commit-and-prove setting, once the witness $\boldsymbol{w}$ is committed, one can compute $\boldsymbol{\Gamma}$ as a random linear combination of $\boldsymbol{\Gamma}_1, \dots, \boldsymbol{\Gamma}_N$ in order to check a system of quadratic equations at once (as outlined in Section 4.4.3 and applied in Protocol 4.4.7), thus reducing as set of quadratic equations to a single quadratic equations, and that one to a rank-1 constraint system.

Observe that the increase in the number of variables seems best possible in the worst case, namely proving an inner product with R1CS. The total number of variables in the R1CS instance is $\max(m,n)$ for $\boldsymbol{w}$ and (at most) $\min(m,n)$ for $\boldsymbol{z}$, hence at most $m + n$.

We recalled Remark B.8.1 because it is surprisingly simple to overlook the approach, especially, if one only needs a few QEs over a few variables. We also stress that there are two different metrics at work.

Remark B.8.2 (The description matters). A problem, described as a rank-1 constraint system may require a lot of (auxiliary) variables, whereas the same problem, described as a quadratic equation system may require very few (auxiliary) variables. Matrix multiplication is one example.[10] In Remark B.8.1 we explained how to use a R1CS in an black-box-like manner to prove the quadratic equation system with small overhead. Proving the (equivalent) R1CS does require the full overhead though, since the very description in terms of R1CS imposes the overhead in terms of auxiliary variables and equations.

To summarize, Remark B.8.1 shows that there is "not too much" overhead to extend proving R1CS to proving general QEs (at least in a commit-and-prove setting). Remark B.8.2 points out in which sense QE can be "more expressive" than R1CS. That is, even though R1CS can prove QEs with little overhead, the *description* certain relations, such as matrix multiplication, is *inherently larger* than the equivalent description in terms of QEs. In that sense, expressivity affects practical efficiency.

[9] This is one equation and variable too much. We can get rid of last equation and variable by using $x_n \cdot (\boldsymbol{\Gamma}\boldsymbol{y})_n + \sum_{i=1}^{n-1} z_i = t$.

[10] Matrix multiplication means asserting lot of inner product constraints. So QEs require no auxiliary variables overhead at all. For R1CS the naive systems of constraints require overhead cubic in the size of the matrix. However, more efficient constraint choices, e.g., constraints based on Strassen's algorithm require fewer multiplications.

Now, we turn to the question of how to verify QEs without going through a R1CS encoding, by looking more closely into the implementation of the argument system. For this, we look into one of the *linearization techniques* used to prove R1CS, and show that it, basically for free, *could prove QEs instead.*

B.8.3. Polynomially Encoded Inner Product Argument

The linearization technique of [AC20] uses (like many others) the Lagrange basis to encode coefficients in evaluation points, and then applies a polynomial identity test to prove that $f \cdot g = h$, where f, g, h are constructed from the coefficients so that this relation corresponds to multiplication triples, i.e. $f(i) \cdot g(i) = h(i)$ for $i = 1, \dots, n$. We ignore the necessity of random terms (i.e. "packed secret sharing") in this note. Consequently, we only describe the idea of encoding R1CS resp. QEs based on linearization. Adding (special) honest-verifier zero-knowledge on top is then a standard exercise, e.g. as in [AC20].

Now, we turn to QEs. First we aim for something simpler, namely an inner product argument (IPA). The IPA is easier to present in the standard basis (instead of the Lagrange bases), and we use $\{X^i\}_{i\in\mathbb{N}_0}$ in $R[X]$ as standard basis. Let $\boldsymbol{a}, \boldsymbol{b} \in R^n$. We want to show that $\langle \boldsymbol{a}, \boldsymbol{b} \rangle := \sum_i a_i b_i = t$.

Consider the product

$$(\sum_{i=0}^{n-1} a_i X^i)(\sum_{j=0}^{n-1} b_j X^{n-j}) = \sum_{k=0}^{2(n-1)} c_k X^k$$

and note that the term $c_n = \sum_{i=0}^{n-1} a_i b_i$ by construction. Thus, to prove that $\langle \boldsymbol{a}, \boldsymbol{b} \rangle = t$, it suffices to know that $c_n = t$. Therefore, in the encoding, simply do not encode c_n, and have it hardcoded to t. Now, a Schwartz–Zippel test allows to check this by using three linear forms to evaluate the above polynomials. Thus, as in [AC20], 3 LPMAs are sufficient (and can be batched down to 1 again).

Now, we extend to QEs, i.e. we encode arbitrary quadratic constraints (or replace $\langle \cdot , \cdot \rangle$ by an arbitrary bilinear form, if you prefer). For this, it is notationally convenient to write $X^{\uparrow n} = (1, X, \dots, X^{n-1})$, $X^{\downarrow n} = (X^{n-1}, \dots, X, 1)$. We want to check $\boldsymbol{a}^\top \Gamma \boldsymbol{b} = \langle \boldsymbol{a}, \Gamma \boldsymbol{b} \rangle = t$. Let $\boldsymbol{\beta} = \Gamma \boldsymbol{b}$ and note that

$$(\sum_{i=0}^{n-1} a_i X^i)(\sum_{j=0}^{n-1} \beta_j X^{n-j}) = \sum_{k=0}^{2(n-1)} c_k X^k \tag{B.8.1}$$

has $c_n = t$ iff $\langle \boldsymbol{a}, \Gamma \boldsymbol{b} \rangle = 0$. And using $\sum_{i=0}^{n-1} a_i X^i = \langle \boldsymbol{a}, X^{\uparrow n} \rangle$ and similarly for $\boldsymbol{b}$ and $\boldsymbol{c}$, we find

$$\begin{aligned} &(\sum_{i=0}^{n-1} a_i X^i)(\sum_{j=0}^{n-1} \beta_j X^{n-j}) = \sum_{k=0}^{2(n-1)} c_k X^k \\ \iff &\langle \boldsymbol{a}, X^{\uparrow n} \rangle \langle \Gamma \boldsymbol{b}, X^{\downarrow n} \rangle = \langle \boldsymbol{c}, X^{\uparrow 2n-1} \rangle \\ \iff &\langle \boldsymbol{a}, X^{\uparrow n} \rangle \langle \boldsymbol{b}, \Gamma^\top X^{\downarrow n} \rangle = \langle \boldsymbol{c}, X^{\uparrow 2n-1} \rangle \end{aligned}$$

Again, these can be checked at a random point which involves 3 linear forms. Note that commitments to $\boldsymbol{a}, \boldsymbol{b}$ and also $\boldsymbol{c}$ are required. For "normal" QEs, we would have $\boldsymbol{a} = \boldsymbol{b} = \boldsymbol{w}$, so only commitments to $\boldsymbol{a}$ and $\boldsymbol{c}$ are required then.

Remark B.8.3. The argument for converting an inner product test to a test for an arbitrary equation is the same as used for $\mathrm{QESA}_{\mathrm{ZK}}$, cf. Section 4.4. Testing multiple quadratic equations, in a commit-and-prove setting setting can be done very efficiently by first committing to $\boldsymbol{a}$ and $\boldsymbol{b}$, and then randomly batching all Γ_i into one Γ which is checked. (Again, this is similar $\mathrm{QESA}_{\mathrm{ZK}}$, but there no term $\boldsymbol{c}$ exists, since an inner product argument is assumed as pivot, unlike the linear map preimage argument in [AC20].)

Remark B.8.4. The folding technique of Bulletproofs uses $\langle \boldsymbol{a}_0 + \boldsymbol{a}_1 X, X\boldsymbol{b}_0 + \boldsymbol{b}_1\rangle = c_0 + c_1 X + c_2 X^2$ as its central identity. When rolling out the recursion with different variables X_i (for round-i challenge), it becomes evident that $\langle \sum_{i\in\{0,1\}^\ell} a_{\vec{i}} X^{\vec{i}}, \sum_{j\in\{0,1\}^\ell} b_{\vec{j}} X^{\vec{j}}\rangle = \sum_{k\in\{0,1,2\}^\ell} c_{\vec{k}} X^{\vec{k}}$ is verified (up to suitable permutation of indices).[11] Importantly, due to the recursion, $\boldsymbol{c}$ need not be known and committed to beforehand (and indeed, it's size is 3^ℓ, that is, quadratic in the length of $\boldsymbol{a}$, $\boldsymbol{b}$). This saves a lot of space. And since $\langle \boldsymbol{a}_i, \boldsymbol{b}_j\rangle \in R$ is a scalar, the c_k's are scalars.[12] Hence they are just two auxiliary variables (as $c_1 = \langle \boldsymbol{a}, \boldsymbol{b}\rangle$ is implicitly known). By modifying the first step in the recursion to $\langle \boldsymbol{a}_0 + \boldsymbol{a}_1 X, \boldsymbol{\beta}_0 + \boldsymbol{\beta}_1 X\rangle$ where $\boldsymbol{\beta} = \boldsymbol{\Gamma}\boldsymbol{b}$, it should be possible to avoid committing to $\boldsymbol{c}$ the above construction as well.[13]

B.8.4. A Special Case

Consider the special case, where one wants to prove a self-inner product $\langle \boldsymbol{w}, \boldsymbol{w}\rangle = t$. In this case, $\boldsymbol{b} = \boldsymbol{a} = \boldsymbol{w}$, so $\boldsymbol{b}$ need not be explicitly committed. For convenience, we change a few encodings and conventions:

- w_0 is a fixed unit, w.l.o.g. $w_0 = 1$ (hence need not be committed to).
- $\boldsymbol{a}(X) = \sum_{i=0}^n w_i X^i$. (Note that we sum to n now, but w_0 is a constant.)
- $\boldsymbol{b}(X) = \sum_{i=0}^n w_i X^{-i}$ (i.e. $\boldsymbol{b}(X) = X^{-n}\boldsymbol{b}'(X)$, where $\boldsymbol{b}'$ is the previous encoding).

Now let

$$\boldsymbol{c}(X) = \boldsymbol{a}(X)\boldsymbol{b}(X) = \sum_{i,j} w_i w_j X^{i-j} = \sum_{k=-n}^{n} X^k \sum_{i-j=k} w_i w_j$$

and observe that $c_i = c_{-i}$ for all $i = 1, \ldots, n$. Thus, it suffices to commit to $c_1, \ldots, c_n$. (Note that $c_0 \langle \boldsymbol{w}, \boldsymbol{w}\rangle = t$ is a constant.)

More generally, consider a quadratic equation $\langle \boldsymbol{w}, \boldsymbol{\Gamma}\boldsymbol{w}\rangle = t$. If $\boldsymbol{\Gamma} = \boldsymbol{G}^\top \boldsymbol{G}$, then $\boldsymbol{c}(X)$ is still symmetric, and $\boldsymbol{G}\boldsymbol{w}$ is a linear transformation of $\boldsymbol{w}$, hence can be applied efficiently in our setting.

Unfortunately, computing this decomposition of $\boldsymbol{\Gamma}$ (if it exists) is probably too expensive, especially in higher dimensions. Moreover, this special case does not seem to generalize further, e.g. $\boldsymbol{c}(x)$ is not symmetric just because $\boldsymbol{\Gamma}$ is symmetric, going through $\langle \boldsymbol{G}\boldsymbol{w}, \boldsymbol{G}\boldsymbol{w}\rangle$ (as described) seems necessary.

B.8.5. The Choice of Languages

We have now seen several aspects in the choice of languages, which are of relevance to and complicate a comparison:

- There is a subtle difference between proving a certain language and reducing one language to another. E.g., reducing QE to R1CS is very efficient (in certain settings) and increases "expressivity".
- Languages have different "expressivity". Intuitively, the more "expressive" a language, the fewer "auxiliary" constructs (variables and constraints) are required to represent a high-level statement. (We have do not offer a formal notion of "expressivity".)

[11] This not only looks like a sumcheck, the work [BCS21]) shows that it may in fact be described as a kind of sumcheck protocol.

[12] For general bilinear forms, e.g. an outer product, this is not true.

[13] The term $\boldsymbol{c}$ in Eq. (B.8.1) has size $2n$, not $3^\ell = n^{\log(3)}$. This is a feature of using univariate polynomials, where the degree of a product is at most the sum of the degrees.

- High-level statements may be encoded differently even in a fixed language, resulting in different efficiency characteristics which is not due to "expressivity", e.g. naive matrix multiplication vs. Strassen's algorithm (Footnote 10).

Example B.8.5. For general R1CS (and also QE), the polynomial $\boldsymbol{c}(X)$ has no obvious structure. However, for arithmetic circuits (AC) each c_i for $i < n$ is itself a wire, and thus half of $\boldsymbol{c}(X)$ is predetermined by $\boldsymbol{w}$, hence does not require additional variables [AC20]. This optimization does not seem to apply to QE-encoded ACs in an obvious manner.

Besides the languages themselves, there is the question of how an argument system deals with its "native" language. Namely, we saw that for R1CS and QE, auxiliary variables are necessary (concretely, the term $\boldsymbol{c}$ in Appendix B.8.3). This makes a simple comparison even more apples-to-oranges. Indeed, as described, our simple QE construction always commits to a degree $2n$ polynomial $\boldsymbol{c}(\vec{X})$, whereas in R1CS the degree of $\boldsymbol{c}(\vec{X})$ scales as $2m$, where m is the number of multiplications (which may be fewer than n).[14] For *naive* arithmetic circuit constructions, this may in fact suggest that R1CS is *better*. Then again, once we start to exploit that we have *equational reasoning*, i.e. we *verify* statements and do *not compute* outputs (as arithmetic circuits must), this "advantage" becomes much less clear. Indeed, inner products and matrix multiplication as statements are examples where QEs seem far superior, both in expressivity and actual efficiency.

Besides the clearly motivated questions above, we can go a bit further, asking: "*What* do we want to prove, and *what is the witness*"? Most importantly, does the witness include auxiliary information or not? E.g. to prove for arithmetic circuit C that $C(x) = y$, one can consider only the input x as the witness, or consider all wire values w as the witness, or choose something in between. Protocols and protocol comparisons depend strongly on such choices, even though all "encode" the same high-level statement.

B.9. Tree-Finding for Short-Circuit-Extraction

In this section, we briefly discuss tree-finding in the setting of short-circuit extraction. We present our candidate short-circuit extractors, whose knowledge error is easily analyzed, but whose runtime analysis appears to more complex.

B.9.1. Generalizing Tree-Finding

Recall that our knowledge extractors for special sound protocols are composed of a tree-finder and tree-extractor, cf. Section 2.4. A useful abstraction was to consider a (deterministic) algorithm A which take as input a sequence of challenges $(\gamma_1, \dots, \gamma_\ell)$ and output 0 or 1. We used the output bit $\mathsf{A}(\gamma_1, \dots, \gamma_\ell)$ to indicate whether or not the challenge sequence would make the verifier accept (when playing with a deterministic prover). This abstraction simplified our discussion, allowing us to hide all details of proof systems and tree-extraction entirely from tree-finding. For short-circuit extraction, we adapt this concept. In this case, it can happen that a tree-extraction fails or short-circuits, and hence the tree (in particular its final shape) must depend on these events. Concretely, after a a subtree was extracted, and μ witnesses $\mathbb{w}_1, \dots \mathbb{w}_\mu$ were obtained for the child nodes, if the (tree-)extractor fails to extract

[14] It is plausible that a similar optimization applies to QE. Perhaps, the QE can be rewritten in the "virtual" coefficients of the multiplication, where by "virtual" we mean any linear combination of "basic" coefficients w_i.

a higher-level witness, then μ' witnesses from child nodes are required to engage in short-circuit extraction.

We abstract failure of quick-extraction by using a predicate φ which takes as input a μ-subtree of challenges and outputs 0 to indicate that this subtree "quick-extracts" successfully, and it outputs 1 to indicate that this subtree requires μ' children (to "short-circuit"). Observe that, given a deterministic prover, the predicate can recompute the witnesses which a tree-extractor would (recursively) obtain and check if quick-extraction would occur or not. Hence, studying tree-finding for a pair (A, φ) of deterministic algorithms is sufficient to handle deterministic provers. Moreover, as with our results for special soundness, extending bounds on deterministic algorithms to probabilistic ones will be straightforward.

As with tree-finding for special soundness, we will use a recursive construction and start by defining and analyzing basic tree-finders.

B.9.2. Basic Tree-Finders

With special soundness, (only) two outputs could occur in a recursive call: Either extraction succeeded or it failed. Hence, it was sufficient to distinguish between success (say, 1) and failure 0. To actually build a tree recursively, we associated an auxiliary string z which contained the extracted subtree to the output, cf. Section 2.5.2. Since associating this auxiliary string is a mere formality necessary to generate the desired output, we will ignore it when describing the basic tree-finders.

Unlike special soundness, we distinguish 4 different outcomes:

- $\top$ indicates a quick success.
- $\overline{\top}$ indicates a short-circuit success.
- $\bot$ indicates an immediate and cheap failure.
- $\underline{\bot}$ indicates an expensive failure.

The distinction between $\bot$ and $\underline{\bot}$ will only become necessary later on, but we introduce it already now. In order to recursively compose our basic tree-finders, we define them for algorithms A which map a challenge $\gamma \in \mathcal{C}$ to an outcome $o \in \{\bot, \top, \underline{\bot}, \overline{\top}\}$.

Our first candidate basic short-circuit extractor is the following modification of $\mathsf{TreeFind}_{\mathsf{NHG}}$ and denoted $\mathsf{QExt}_{\mathsf{NHG}}$. By construction, $\mathsf{QExt}^{\mathsf{base}}_{\mathsf{NHG}}$ immediately returns $\overline{\top}$ whenever it is encountered — this is the idea of short-circuiting. Otherwise, $\mathsf{QExt}^{\mathsf{base}}_{\mathsf{NHG}}$ simply samples without replacement, and after finding μ accepting challenges checks if quick-extraction succeeds. If not, $\mathsf{QExt}^{\mathsf{base}}_{\mathsf{NHG}}$ continues sampling until μ' accepting challenges are found and if so, outputs $\overline{\top}$ to indicate short-circuiting. Observe that $\mathsf{QExt}^{\mathsf{base}}_{\mathsf{NHG}}$ outputs $\underline{\bot}$ if extraction failed but at least μ accepting transcripts were found — this is an "expensive" failure. Formally, $\mathsf{QExt}^{\mathsf{base}}_{\mathsf{NHG}}$ is defined as follows:

1. Pick $\gamma_1 \xleftarrow{\$} \mathcal{C}$ uniformly. Let $o \coloneqq \mathsf{A}(\gamma_i)$.
2. If $o = \bot$ return $\bot$.
3. If $o = \underline{\bot}$ return $\underline{\bot}$.
4. If $o = \overline{\top}$ return $\overline{\top}$. Else let $i = 2$.
5. Repeat until $i > \mu$ or all of $\mathcal{C}$ was exhausted.

```
        a) Pick $\gamma_i$ uniformly from $\mathcal{C}$ without replacement.
        b) Let $o \coloneqq \mathsf{A}(\gamma_i)$.
        c) If $o = \top$, increase $i$ to $i + 1$.
        d) If $o = \overline{\top}$, return $\overline{\top}$.
    6. If $i \leq \mu$, return $\bot$.
    7. If $\varphi(\{\gamma_1, \dots, \gamma_k\}) = 0$ return $(\gamma_1, \dots, \gamma_k)$
    8. Repeat until $i > \mu'$ or all of $\mathcal{C}$ was exhausted.
        a) Pick $\gamma_i$ uniformly from $\mathcal{C}$ without replacement.
        b) Let $o \coloneqq \mathsf{A}(\gamma_i)$.
        c) If $o = \top$, increase $i$ to $i + 1$.
        d) If $o = \overline{\top}$, return $\overline{\top}$.
    9. If $i \leq \mu'$, return $\underline{\bot}$. Else return $\overline{\top}$.
```

Note that by construction, $\mathsf{QExt}^{\mathsf{base}}_{\mathsf{NHG}}$ effectively treats $\bot$ and $\underline{\bot}$ the same. When the basic tree-finder $\mathsf{QExt}^{\mathsf{base}}_{\mathsf{NHG}}$ is composed recursively, it yields our first candidate short-circuit extractor $\mathsf{QExt}_{\mathsf{NHG}}$. It is straightforward to see that $\mathsf{QExt}_{\mathsf{NHG}}$ for $(\mu_1, \dots, \mu_\ell)$-quick $(\mu'_1, \dots, \mu'_\ell)$-short extractability has knowledge error at most that of $\mathsf{TreeFind}_{\mathsf{NHG}}$ for $(\mu'_1, \dots, \mu'_\ell)$-special soundness. While this may not be fully optimal in the face of short-circuit extraction, it is the same as the knowledge error when only special soundness is considered.[15] Unfortunately, we do not have a provable bound on the expected runtime of $\mathsf{QExt}_{\mathsf{NHG}}$. Nevertheless, it is we find it plausible that its expected runtime is bounded by the maximal number M of required transcripts (cf. Corollary 4.2.19).

To provide a more credible candidate, we consider a modified extractor $\mathsf{QExt}^{\mathsf{base}}_{\mathsf{early}}$ and its recursive composition $\mathsf{QExt}_{\mathsf{early}}$. This modification to $\mathsf{QExt}_{\mathsf{NHG}}$ introduces early exits, namely, it immediately returns when $\underline{\bot}$ is encountered. It would be very surprising if $\mathsf{QExt}_{\mathsf{early}}$ would exceed an expected number of M queries to A, where M is as above. Indeed, heuristically, one can apply the same argument as used in Corollary 4.2.19 to bound the runtime; technically, difficulties arise and the analysis become more complex than the intuition suggest.

Definition B.9.1. $\mathsf{QExt}^{\mathsf{base}}_{\mathsf{early}}$ is defined as $\mathsf{QExt}^{\mathsf{base}}_{\mathsf{NHG}}$, except for the following change: Whenever $\underline{\bot}$ is encountered, $\mathsf{QExt}_{\mathsf{early}}$ immediately returns $\underline{\bot}$. That is, after lines "if $o = \underline{\bot}$ return $\underline{\bot}$", the line "if $o = \underline{\bot}$ return $\underline{\bot}$" is appended.

The recursive composition, $\mathsf{QExt}^{\mathsf{base}}_{\mathsf{NHG}}$ is defined as in Section 2.5.2.

[15] We have seen examples where protocols are $(\mu' - 1)$-special sound, but μ'-short extractable, leading to a small gap, e.g. Lemma 4.3.19. Taking this into account, it may be possible to construct an even better extractor. For example, it may be beneficial to immediately output a witness when it can be computed from a subset of accepting challenges, instead of waiting until μ' accepting challenges were found (if quick-extraction failed). However, success and runtime analysis of such protocols becomes even more complicated.

The remainder of the section is to analyze the knowledge error of $\mathsf{QExt}_{\mathsf{early}}$. We will see that for $(\mu_1, \dots, \mu_\ell)$-quick $(\mu'_1, \dots, \mu'_\ell)$-short extractability where the cardinality $\#\mathcal{C}_i$ of all challenge sets is lower-bounded by N, $\mathsf{QExt}_{\mathsf{early}}$ has a knowledge error of roughly $\prod_{i=1}^{\ell} \mu'_i / N$. While this is far from optimal, it is often still reasonable. For example, with $\mathsf{QESA}_{\mathsf{ZK}}$ or Bulletproofs in 256-bit groups with full-sized challenges, the term is roughly $n^4/2^{256}$ where n is the witness dimension. Hence, for $n = 2^{32}$ there is still a provable knowledge error of roughly 2^{-128} for $\mathsf{QExt}_{\mathsf{early}}$, while the number of queries to A is reduced from roughly $n^4 = 2^{128}$ to heuristically $n \log(n) \leq 2^{40}$.

B.9.3. Success Analysis of $\mathsf{QExt}_{\mathsf{early}}$

Before starting with the analysis, we record following result.

Lemma B.9.2. *Let $m, n, k \in \mathbb{N}$ and $m \geq k$. Then*

$$1 - \prod_{i=0}^{k-1} \frac{m-i}{m+n-i} = 1 - \prod_{j=1}^{n} \left(1 - k\frac{1}{m+j}\right) \leq k \cdot \frac{n}{m+n}$$

We postpone the proof until Appendix B.9.3.1 and continue with the success analysis of $\mathsf{QExt}_{\mathsf{early}}$. Following Lemma assures that the abort probability due to $\underline{\perp}$ only grows linearly in the size of the $\boldsymbol{\mu}$-tree,i.e. in $\prod_{i=1}^{\ell} \mu_\ell$.

Lemma B.9.3. *Let Π be a $(\mu'_1, \dots, \mu'_\ell)$-quick $(\mu_1, \dots, \mu_\ell)$-short extractable argument system with challenge spaces size $(N_1, \dots, N_\ell)$ and let $\mathsf{QExt}^{\mathsf{base}}_{\mathsf{early}}$ be as in Definition B.9.1 and $\mathsf{QExt}_{\mathsf{early}}$ its recursive composition. Let $N = \min(N_1, \dots, N_\ell)$. We call a (deterministic) algorithm A ordinary if it only outputs $\perp$ and $\top$. Then for any (deterministic) ordinary algorithm A, we have*

$$\Pr[\mathsf{QExt}^{\mathsf{A}}_{\mathsf{early}} = \underline{\perp}] \leq \frac{\mu_1 \cdot \ldots \cdot \mu_\ell}{N}$$

Proof. As noted before Appendix B.9.1, we can interpret extraction and A as finding enough leafs with $\mathsf{A}(\gamma_1, \dots, \gamma_\ell) = \top$ in a tree of challenges. On the other hand, if $\mathsf{QExt}_{\mathsf{early}}$ encounters a single $\underline{\perp}$ (resp. $\overline{\top}$) during its recursion, it immediately completes the entire extraction with output $\underline{\perp}$ (resp. $\overline{\top}$). To prove the claim by induction over ℓ, the number of challenges, we first show following claim, which will serve as both base case and induction step.

Claim B.9.4. *First, we consider $\mathsf{QExt}^{\mathsf{base}}_{\mathsf{early}}$, i.e. $\ell = 1$ and $\mu = \mu_1$, $N = N_1$. Let A an arbitrary (i.e. not necessarily ordinary). In that case, we get*

$$\Pr[\mathsf{QExt}^{\mathsf{A}}_{\mathsf{early}} = \underline{\perp}] \leq \mu \cdot \Pr_{\gamma \xleftarrow{\$} \{1,\dots,N\}}[\mathsf{A}(\gamma) = 1].$$

Proof. Let $n_\perp$, $n_{\underline{\perp}}$, $n_\top$, $n_{\overline{\top}}$ denote the number of challenges which yields this output, i.e. $\Pr[\mathsf{A}(\gamma) = t] = n_t/N$. Observe that the success probability only worsens if $n_{\overline{\top}} = 0$ (i.e. if we simply treat $\overline{\top}$ as $\top$). Hence, we consider a setting with only challenges of type $\perp$, $\underline{\perp}$ and $\top$, where a challenge γ has type t if $t = \mathsf{A}(\gamma)$. Moreover, as a first step, we suppose $n_\perp = 0$, i.e. all outputs are either $\underline{\perp}$ or $\top$. In this situation, the extraction will draw (at most) k challenges, and succeed if and only if none of the challenges are $\underline{\perp}$. Hence in this case we get

$$\Pr[\mathsf{QExt}^{\mathsf{A}}_{\mathsf{early}} = \underline{\perp}] = 1 - \Pr[\mathsf{QExt}^{\mathsf{A}}_{\mathsf{early}} \neq \underline{\perp}] = 1 - \prod_{i=1}^{\mu} \frac{n_\top - i}{n_\top + n_{\underline{\perp}} - i} \leq \mu \cdot \frac{n_{\underline{\perp}}}{n_\top + n_{\underline{\perp}}} = \mu \cdot \Pr[\mathsf{QExt}^{\mathsf{A}}_{\mathsf{early}} = \underline{\perp}].$$

where we also use Lemma B.9.2 for the inequality.

Now, we argue how challenges of type $\bot$ affect this. Either the first challenge is $\bot$, in which case extraction outputs $\bot \neq \underline{\bot}$, or it is not $\bot$, in which case we are in the same setting as above since challenges of type $\bot$ are effectively ignored by the extractor (it simply draws another challenge). Let $I = \mathsf{A}(\gamma_1)$ be the type of the first challenge. Then we find:

$$\begin{aligned}
\Pr[\mathsf{QExt}^{\mathsf{A}}_{\mathsf{early}} = \underline{\bot}] &= \Pr[I \neq \bot] \cdot \Pr[\mathsf{QExt}^{\mathsf{A}}_{\mathsf{early}} = \underline{\bot} \mid I \neq \bot] \\
&\leq \frac{n_\top + n_{\underline{\bot}}}{N} \cdot \left(\mu \cdot \frac{n_{\underline{\bot}}}{n_\top + n_{\underline{\bot}}}\right) \\
&= \mu \cdot \Pr[I = \underline{\bot}]
\end{aligned}$$

□

With Claim B.9.4 at hand, we can prove the Lemma by induction.

Base case ($\ell = 1$). In the base case, $\ell = 1$, the claim follows since no output of an ordinary A leads to $\underline{\bot}$. Indeed, $\underline{\bot}$ and $\overline{\top}$ are only introduced by $\mathsf{QExt}_{\mathsf{early}}$ in non-leaf nodes; Claim B.9.4 considered generalized A. Thus, in this case we know that to output $\underline{\bot}$ can only occur if there are at most $\mu_1 - 1$ challenges with $\mathsf{A}(\gamma_1) = \top$. Hence $\Pr[\mathsf{QExt}^{\mathsf{A}}_{\mathsf{early}} = \underline{\bot}] \leq \frac{\mu_1 - 1}{N_1} \leq \frac{\mu_1}{N}$ as claimed.

Induction ($\ell - 1$ to ℓ). By definition $\mathsf{QExt}_{\mathsf{early}}$ is a recursive application of the basic extractor applied to a A. By induction hypothesis, after extracting $(\ell - 1)$ rounds, the obtained A' (generated by the recursive applications of $\mathsf{QExt}^{\mathsf{base}}_{\mathsf{early}}$, cf. Section 2.5.2) satisfies:

$$\Pr[\mathsf{A}'(\gamma_\ell) = \underline{\bot}] \leq \prod_{i=1}^{\ell-1} \mu_i \cdot \Pr[\mathsf{A}(\gamma_1, \ldots, \gamma_\ell) = \underline{\bot}].$$

By Claim B.9.4, the induction hypothesis extends to ℓ. This completes the proof of Lemma B.9.3. □

Corollary B.9.5. *Consider the same setting as in Lemma B.9.3. Then the knowledge error of* $\mathsf{QExt}_{\mathsf{early}}$ *is bounded by*

$$\sum_{i=1}^{\ell} \frac{\mu_i}{N_i} + \frac{\mu_1 \cdot \ldots \cdot \mu_\ell}{N}$$

Proof sketch. Let E be the event that $\mathsf{QExt}^{\mathsf{A}}_{\mathsf{early}}$ outputs $\underline{\bot}$. Conditioned on $\neg E$, i.e. E not happening, $\mathsf{QExt}^{\mathsf{A}}_{\mathsf{early}}$ behaves like $\mathsf{TreeFind}_{\mathsf{NHG}}$, except that a few children are skipped. Since $\sum_{i=1}^{\ell} \frac{\mu_i}{N_i}$ is a (rough) bound for the knowledge error of $\mathsf{TreeFind}_{\mathsf{NHG}}$ (cf. Section 2.5.2) and since $\Pr[E] = \Pr[\mathsf{QExt}^{\mathsf{A}}_{\mathsf{early}} \neq \underline{\bot}] \leq \frac{\mu_1 \cdot \ldots \cdot \mu_\ell}{N}$ by Lemma B.9.3, the claim follows. □

B.9.3.1. Proof of Lemma B.9.2

Proof. The first equality follows by following equations

$$\begin{aligned}
\prod_{i=0}^{k-1} \frac{m-i}{m+n-i} &= \frac{\prod_{i=0}^{k-1}(m-i)}{\prod_{i=0}^{k-1}((m+n)-i)} = \frac{m!/(m-k)!}{(m+n)!/(m+n-k)!} \\
&= \frac{(m+n-k)!/(m-k!)}{(m+n)!/m!} = \frac{\prod_{j=1}^{n}(m+j-k)}{\prod_{j=1}^{n}(m+j)} = \prod_{j=1}^{n}\left(1 - k\frac{1}{m+j}\right)
\end{aligned} \tag{B.9.1}$$

With this rewriting, the claim for $n = 1$ is immediate.

Claim B.9.6. *The claim holds for $n = 1$ (and any $m \geq k$).*

Let (for arbitrary but fixed m, k)

$$f(n) = 1 - \prod_{i=0}^{k-1} \frac{m-i}{m+n-i}$$
$$g(n) = k \cdot \frac{n}{m+n}$$

We have to show $f(n) \leq g(n)$ for arbitrary m, n, k. By Claim B.9.6, the claim holds for $n = 1$. To extend it to $n \geq 1$, we could show that $\frac{\partial f}{\partial n} \leq \frac{\partial g}{\partial n}$ for $n \geq 1$. Instead, we show it for discrete steps, i.e. we show following claim:

Claim B.9.7. *For $n \geq 1$ and $m \geq k$, we have*

$$f(n+1) - f(n) \leq g(n+1) - g(n)$$

Clearly, if Claim B.9.7 holds, then in particular $f(n) \leq g(n)$ implies $f(n+1) \leq g(n+1)$. Hence, the lemma follows by induction using the base case $f(1) \leq g(1)$ recorded in Claim B.9.6.

Now we prove Claim B.9.7. Let $\pi_n = \prod_{j=1}^{n}(1 - k\frac{1}{m+j})$ (as in Eq. (B.9.1)) Then we have to show hat

$$f(n+1) - f(n) = 1 - \pi_{n+1} - (1 - \pi_n) = \pi_{n+1} - \pi_n = \pi_n \frac{k}{m+n+1}$$

is upper bounded by

$$g(n+1) - g(n) = k\left(\frac{n+1}{m+n+1} - \frac{n}{m+n}\right) = k\frac{m}{(m+n)(m+n+1)}.$$

We can simplify this to

$$\pi_n \frac{k}{m+n+1} \leq k\frac{m}{(m+n)(m+n+1)} \qquad \Longleftrightarrow \qquad \pi_n \leq \frac{m}{m+n}$$

Finally, switching to the equivalent product of π_n in Eq. (B.9.1) and using that each factor is at most 1, we observe that

$$\pi_n = \prod_{i=0}^{k-1} \frac{m-i}{m+n-i} \leq \frac{m}{m+n}.$$

This concludes the proof of Claim B.9.7 and hence the lemma. □

C. Appendix for Chapter 5

C.1. Machine Models

We do not want to go into much detail about the machine model, and will essentially assume that it is admissible. Admissibility carries certain explicit semi-formal requirements. As our machine model, we have some RAM-like model in mind. Indeed, "concrete efficiency" is relatively important when dealing with *expected* time. Recall that there are (runtime) distributions T over $\mathbb{N}_0$ with $\mathbb{E}[T] < \infty$ but $\mathbb{E}[T^2] = \infty$. Thus, we require that certain operations can be carried out efficiently (e.g. with logarithmic overhead). Importantly, we require efficient arithmetic and the abilitiy to use standard efficient construction, such as arrays or more sophisticated *data structures*, which allow efficient computation in a RAM model (or *multi-tape* Turing machine). We also require *efficient emulation* of (efficient) programs, oracles, or interactive systems in the sense that "emulating" an execution does not affect the runtime too much. Moreover, emulation allows to truncate, suspend, resume, rewind, or similarly affect executions based on efficiently computable events (such as the number of steps emulated, or messages received).

Remark C.1.1 (Non-Halting). Non-halting computations are an irksome technical artefact. To deal with them explicitly, we define the symbol `nohalt` as the "output" of such a computation, and assume that any system which receives `nohalt` also outputs `nohalt`, if not specified otherwise. Alternatively, one can follow [Gol10] and assume all algorithms halt after a finite number $n(\lambda)$ of steps. This introduces (arbitrarily) small deviations for "perfectness", e.g. it is again impossible to sample from $\mathsf{Ber}(1/3)$.

C.1.1. Systems, Oracles, Algorithms

Before considering machine models and specific properties, we sketch the high level abstractions. We view algorithms and oracles as systems, which offer (communication) interfaces. Interfaces allow to receive and/or send messages. For example, the input (resp. output) interface typically receives (resp. sends) exactly one message, the input (resp. output). To model "laziness", one may view the interface less strictly, and allow the input (resp. output) interface to read symbol for symbol. Thus, a calling algorithm need not provide the full input (resp. output) at once. This is relevant in our setting, where input (resp. output) lengths are not a priori bounded.

We do not formalize the means of interfacing precisely, but argue in a hand-wavy manner. (In our case, with many competing definitions of machine and communication models, we believe it is better to be explicitly imprecise, than importing a lot of unnecessary details.)

We work with three related notions: *Systems*, *oracles*, and *algorithms*. A **deterministic system** is defined by its interfaces and "input-output behaviour" only, i.e. it is a "mathematical object". A

(probabilistic) system is a random variable S, such that any realization of S is a deterministic system.[1] A system has no notion of “runtime”, or “random tape”. By connecting interfaces, systems may interact. This forms a new system. Any system has an implicit input, the security parameter. A system is **closed** if the only input is its security parameter, and it offers only an output interface.

An **algorithm** is given by *code* (perhaps non-uniformly) and bound to a *machine model.* The code and machine model describe its behaviour as a system, and impart it with a notion of runtime and “random tape”. (Randomness need not be modelled by a random tape.)

By **oracle** or **party**, we denote systems or algorithms to which only interface access is used. For example, black-box rewinding access (bb-rw) to an adversary means access to an oracle (with an underlying algorithm in this case). If not indicated otherwise, an oracle $\mathbb{O}$ is an algorithm (to which only interface access is provided).

In our setting, a convenient abstraction are **timed** oracles, which allow execution for an a priori *bounded time*, and which *report the elapsed time* to the caller when answering a query (or report `timeout`, if it did not complete in time). See Appendix C.1.3 for a more precise specification. Timed bb-rw simulators can make use of this to truncate overlong executions, and this corresponds to *extended black-box access* in [KL08].

Another useful abstraction, mostly for convenience in the setting of a posteriori efficiency, are **timeful** oracles (or timeful systems). **Timeful** oracles are systems, which provide a *purported elapsed runtime* to the machine model. Importantly, timeful oracles are not bound by complexity notions or machine models, except satisfying consistency restrictions, e.g. their purported runtime must be long enough to have written the answer to the interface. Hardness assumptions, such as timelock puzzles are void against timeful oracles. Thus, they are a means to formalize unconditional runtime guarantees for algorithms with oracle-access, e.g. bb-rw simulators, but also serve as a convenient abstraction, e.g. for Lemma 5.3.12. A timeful oracle also yields a timed oracle in the obvious way.

C.1.2. Abstract Machine Model Operations and Interaction

From an abstract point of view, we want a machine model with following properties:[2]

Efficient arithmetic which does not thwart our results.

Efficient data structures such as arrays (i.e. random access), or something morally equivalent.

Abstract subroutines such as oracle calls, or a message sending function.

Abstract access to subroutine results. This is non-trivial, in particular if subroutines need not be efficient. Thus, even for a RAM-model, accessing the result of an oracle needs some tape-like access method.[3]

[1] Our definition of system is ad-hoc. A compatible, precise notion was recently (concurrently) introduced in [LM20]. We allow two probabilistic systems to behave identically, whereas in [LM20] equivalence classes are considered (and what we call system is called “probabilistic discrete system”). We prefer to work with concrete representatives, as having a concrete probability space at hand significantly simplifies definitions and reasoning, though it is not strictly necessary.

[2] Another requirement, which is natural enough that we did not prominently require it, is that to send message of length n, some time is required. We assume n steps for length n as a lower bound.

[3] The problem here is: If the result of an oracle is *huge*, any access may exhaust the alloted runtime. This is nonsense (and completely breaks our results). For that reason, some (trivial, efficient) encoding for such unbounded objects are necessary, e.g. bitwise tape-like. Concretely, our runtime oracles might output gigantic runtimes, which a runtime distinguisher need not completely read to discern them from polynomial time.

Interactive machines which communicate and are activated in some sensible way.

Efficient emulation ensures that one can efficiently execute code and emulate many interacting algorithms with little overhead.

A notion of runtime which is *local*, i.e. one can separate between time spent within some machine, subroutine, or oracles, and account accordingly.

Let us formalize our wishes a bit. Concerning arithmetic and data structures, we want typical algorithms to be efficient. In particular, distinguishing distributions by sampling often enough and computing the empirical distribution should be "efficient" in the sample size n, see Appendix C.3.4. For data structures, we may have to deal with excessively large inputs, thus, we may need suitable encodings, e.g. a tuple should allow access to any of its components efficiently, even with tape-like access. For example, representing (x, y) by concatenation only works if x is guaranteed to be short, but is inefficient if x is very long. Interleaving always works for tuples of constant dimension.

Now, we formalize the locality of runtime. Let $\mathsf{A}^{\mathcal{O}_1,\dots,\mathcal{O}_N}$ be an oracle machine (with access to N oracles). We require that

$$\mathsf{time}_{\mathsf{A}+\mathcal{O}_1+\mathcal{O}_2+\dots}(\mathsf{A}^{\mathcal{O}_1,\dots,\mathcal{O}_N}) = \mathsf{time}_{\mathsf{A}}(\mathsf{A}^{\mathcal{O}_1,\dots,\mathcal{O}_N}) + \mathsf{time}_{\mathcal{O}_1}(\mathsf{A}^{\mathcal{O}_1,\dots,\mathcal{O}_N}) + \mathsf{time}_{\mathcal{O}_2}(\mathsf{A}^{\mathcal{O}_1,\dots,\mathcal{O}_N}) + \dots \quad \text{(C.1.1)}$$

though "morally equivalent" relaxation suffice for most results. (Note that our algorithm takes no input. In case of randomized algorithms, the runtimes for A, $\mathcal{O}_1, \dots \mathcal{O}_N$ are *not* stochastically independent.)

Finally, a sensible machine model guarantees efficient emulation. Namely, if $\mathsf{time}_{\mathsf{A}+\mathcal{O}_1+\mathcal{O}_2+\dots}(\mathsf{A}^{\mathcal{O}_1,\dots,\mathcal{O}_N})$ is efficient so is the runtime of the algorithm $\mathcal{B}$ which *emulates* the execution of all oracles. In other words, converting an (interacting) system of machines into a single machine $\mathcal{B}$ by emulating all parties (or oracles) *preserve efficiency*. Furthermore, emulation should efficiently allow to gather (and act upon) execution statistics, most importantly the elapsed runtime of the emulated code, and the possibility to truncate an oracle emulation after a number of steps. Emulation should behave just like one expects from a virtual machine, in particular, be possible step-for-step.

Note that preservation of efficiency depends on the machine model and the notion of efficiency itself. For example, if emulation has a logarithmic overhead, then linear time is not preserved under emulation, but quasi-linear time may be. Emulation overhead which is linear (or better sublinear) in the number of emulated steps is a very convenient property of a machine model. We write $\mathsf{emuovhd}_{\lambda,N}(k)$ for the time steps required to emulate k steps (of a N machine/oracle system in some implicit machine model). Usually, the security parameter λ and number of oracles N are suppressed.

The Communication Model. We will assume an communication model where messages of arbitrary size can be sent, and parties have incoming messages queues. These do not count towards their space, and they do not pay runtime for receiving a message, only for reading it. Tape-like access to messages seems most natural, so we assume that. For technical reasons, one may wish provide the possibility of dropping (i.e. skipping) a (partially read) message. This allows a party to ignore large messages, keeping its runtime in check. Another possibility is to use fixed size messages (packages), and make the transfer of longer messages an "explicit" protocol. With this approach, our simplified view of "inputs as messages" is broken. This surfaces a technical detail, namely that reading from tapes and interacting with an interface which provides the same information is essentially the same, but technically different. By suitably restricting adversaries and algorithms, or introducing "unidirectional channels" (e.g. dummy transmitter parties) for passing inputs (after termination), this can be reconciled.

There are also different strategies for dealing with messages from super-constantly many parties, e.g. one tape-like message queue for all, one message queue per party, etc. Since our focus is (essentially) a two-party setting, we leave technical details, problems, solutions and their relations to the reader.

Non-Uniformity. For non-uniform machines, we propose an advice interface just like the randomness interface.[4] That is, the advice string has infinite length. This seems to be the most natural choice from a machine-model perspective. Complexity classes can then restrict access further, e.g. to expected or strict polynomial size advice. Note that non-uniformity comes with its own more or less subtle anomalies, see e.g. [KM13].

C.1.3. Timed Black-Box Emulation with Rewinding Access

We define *(timed) black-box emulation* similar to [KL08], which differs from standard black-box emulation essentially by making the "runtime/instruction counter" part of the visible black-box interface and by allowing runtime truncation.

Definition C.1.2 (Timed black-box emulation with rewinding access (bb-rw)). A **black-box emulation** oracle $\mathcal{O}$ gives oracle access to a "virtual machine" running some (once and for all) specified program/code. The code may involve multiple (abstracted) parties. Unless otherwise specified, $\mathcal{O}$ behaves *deterministically* in the sense that the randomness of the emulated programs is sampled and fixed prior to interaction.[5] We do *not* let the caller choose the randomness.

The black-box interface depends on the specific type.

- **Fully** black-box emulators take an input message m and return their program's answer a.
- **Timed** black-box emulators take a pair (m, t), where t is a maximum time bound, and return a pair (a, s), where s is the number of steps emulated. If s would exceed the alloted time t, the emulation is aborted and `timeout` is returned. A time bound of $t = \infty$ is allowed. (Execution may be resumed after `timeout`.)
- Black-box emulation **with rewinding access** (bb-rw) allows the state of the emulated program to be stored and loaded. While, a state is identified by its partial transcript of (previous) queries, other means of identification, such as handles, are used to ensure efficiency. Loading, storing, and deleting program states is done by special types of messages.[6]

Note that we distinguished black-box oracles with rewinding access from "normal" oracles. The reason is that the "next-message" approach usually used to implement black-box access is not efficient enough with expected time.

Example C.1.3 (Runtime squaring for $\mathsf{NextMsg}$). Consider following interaction $\langle \mathsf{A}(n), \mathsf{B}\rangle$: First A sends n to B. Then A pings B n times, each times B returns a secret, which A uses in the next ping. Obviously, this interaction runs in time $O(n)$. Consider a distributions N of inputs n on $\mathbb{N}$ with the property that

[4] The advice interface should follow the same restrictions as the "random tape' (see Remark C.1.6), in particular it should not provide memory to not conflate advice complexity with space complexity.

[5] When such an oracle is implemented, the "random tape" (or the respective notion in the machine model) is sampled (and fixed) lazily, just like a random oracle.

[6] Note that all of the code and interfaces which are in our control, e.g. the interface of the black-box are assumed to be nice and well-typed.

$\mathbb{E}[N] < \infty$ but $\mathbb{E}[N^2] = \infty$. Then emulation with next-message-function $\mathsf{NextMsg}$ is not efficient. The reason is that $\mathsf{NextMsg}$ always (re)computes from scratch, which needs about $\sum_{i=1}^{n} i \approx \frac{1}{2}n^2$ steps.

Fortunately, most problems like this arise from repeated computations (or repeated copying) being expensive, and are solved by making recomputing (or copying) superfluous. Computations can be cached, as done in bb-rw implementations. Copying can be reduced by sharing memory access, or passing around access to a machine or interface which implements such a shared memory access.

Remark C.1.4 (Cached UID $\mathsf{NextMsg}$ access). Caching (all) visited states and using short unique identifiers (UID) for visited states (instead of resending the history of messages leading to a state), yields a $\mathsf{NextMsg}$-like function which is a suitable bb-rw oracle implementation (in all situations we have tried). Cached state and short UIDs prevent the quadratic computational overhead, but require *expected* polynomial space. Judiciously caching only important states is typically possible, so that usually strict polynomial space solutions exist.

Keeping track of identifiers and the rewinding tree can be done with efficient data structures. (Polylogarithmic overhead is admissible by Corollary C.1.8.)

Remark C.1.5. For admissible models, emulation of algorithms allows (efficient) runtime cutoffs. Cloning a machine's state, and resuming from a given state should also be (efficiently) possible. (Or we may add it as an new assumption.)

Remark C.1.6 (Space overhead). We have only considered *time* overhead of emulation. This is justified, as it bounds the space/memory overhead. However, memory overhead is an interesting quantity on its own. For example, one might argue that *expected poly-time (EPT)*, but *strict poly-space (SPS)*, is a "more natural" class of feasible computation than expected poly-time and expected poly-space.[7]

While SPS seems to prevent many technical artefacts, it unveils certain others. Depending on the implementation of the randomness interface (e.g. input, read-only tape, coin-toss, …) emulation and bb-rw oracle implementations may not be SPS, because space and randomness complexity are mixed. If read-only access to an (infinite) random tape is given, then emulating two such tapes by "splitting" one works well. If randomness is a coin-toss interface, which upon invocation returns a fresh random bit, then emulation still works. However, to implement a bb-rw oracle $\mathsf{bbrw}(\mathcal{O})$, which gives access to $\mathcal{O}$ with *fixed* randomness, requires to remember all used randomness. This can require expected polynomial space.

How this can be resolved elegantly is an interesting question. One could rely on derandomization, e.g. with an (a priori PPT) pseudorandom function, to simulate a long enough random string with small space. Alternatively, one could try to work with probabilistic bb-rw oracles, which, when rewound to a state use fresh randomness for new queries, i.e. the same query may yield different answers. Our problem with deterministic versus probabilistic access seems related to [BG11].

Similar to randomness, non-uniform (infinite) advice may need to be saved by a bb-rw implementation, leading to space overhead. Again, it depends on the concrete modelling.

[7] Of course, the actual complexity class of interest allows EPT-SPS violation with negligible probability.

C.1.4. (Probably) Admissible Machine Models

To the best of our knowledge, RAM models, and also multi-tape Turing machines, are admissible if one works with polynomial time or larger runtime classes.[8] Following trivial lemma is useful to see that efficient emulation is not hard to achieve, even for *expected* time.

Lemma C.1.7. *Let $f\colon \mathbb{N}_0 \to \mathbb{N}_0$ be any (monotone) strictly increasing function with (monotone) increasing left-inverse g, i.e. $g \circ f = \mathrm{id}$ (but not necessarily $f \circ g = \mathrm{id}$). Suppose T is a runtime and smaller than f, i.e. $\Pr[T_\lambda > f(\lambda)] = 0$ (for all λ). Let h be another monotone function. Then $\mathbb{E}[h(g(T_\lambda))T_\lambda] \le h(\lambda)\mathbb{E}[T_\lambda]$.*

Proof. Use $h(g(T_\lambda)) \le h(g(f(\lambda))) \le h(\lambda)$. □

Corollary C.1.8. *Let poly be any monotone polynomial, and $\mathbb{E}[T_\lambda] \le t(\lambda)$ for a polynomial bound t, and $T \le 2^\lambda$. Then $\mathbb{E}[\mathrm{poly}(\log(T_\lambda))T_\lambda]$ is polynomially bounded (namely by $\le \mathrm{poly}(\lambda)t(\lambda)$).*

Proof. Use Lemma C.1.7 with $f(\lambda) = 2^\lambda$, $g(\lambda) = \log_2(\lambda)$. and $h = \mathrm{poly}$, □

Note that $T_\lambda \le 2^\lambda$ is easily achieved via a runtime cutoff after 2^λ steps.[9] This induces a statistically negligible change in the output of any expected polynomial time algorithm.[10] Thus, we see that polylogarithmic multiplicative overhead in emulation is not a problem for expected polynomial time computations. By taking a smaller superpolynomial bound, e.g. $f(\lambda) = \lambda^{\log(\lambda)}$, we get we a bit more freedom in the emulation overhead.

Remark C.1.9 (Interaction of Corollary C.1.8 and virtuality). CEPT and CPPT ignore negligible events, because they can be hidden in the virtuality. So, Corollary C.1.8 may always be applied after conditioning on the event $\{T_\lambda \le 2^\lambda\}$, i.e. after using the “virtuality slack”. Consequently, polylog overhead is not a problem for CEPT.

We end our discussion of machine models by taking a closer look two exemplary machine models.

Example C.1.10 (Single-tape Turing machines are not admissible). Consider single-tape Turing machine as the model of computation. It is easy to construct an *interactive* algorithm for computing whether a string is a palindrome which runs in *linear* time (in the length of the input string). However, it is well-known that single-tape Turing machines need quadratic time to recognize this language. Thus, the emulation overhead is (at least) quadratic. Hence, it is single-tape Turing machines are not an admissible model of computation.

Example C.1.11 (RAM models). Various RAM models seem appropriate for our cause. A model of computation in which the RAM’s word size grows with the “problem size” seems particularly well-suited for cryptography; indeed security parameter λ is a natural measure for the “problem size”.

[8] We have not carried out formal proofs.
[9] Technically, we have to do an earlier cutoff, since emulation and cutoff also consume runtime. But this is a minor issue.
[10] Unfortunately, such a truncation can affect *perfect* properties, such as perfect correctness, leading to technical artefacts.

C.2. Supplementary Definitions for Commitment Schemes

The commitment scheme used in [GK96] does not fit directly into our notion of non-interactive commitments with setup (Section 2.2.4), since it is in the plain model. Moreover, in the preliminaries, we only defined security of commitment schemes against a priori PPT adversaries. In this section, we explain the small changes need to extend the definitions.

C.2.1. 3-move Commitments Without Trusted Setup

We briefly discuss 3-move commitment schemes in the plain model.

Remark C.2.1 (Plain-model 3-move Commitment Schemes). Our definitions assume that the commitment key is set up by a trusted party. One can obtain almost-non-interactive commitments in the plain model, by allowing the receiver or the committer to set up the commitment key. In this case, one needs an additional algorithm VfyCK for verifying the well-formedness of a (purported) commitment key ck, since it could now be (maliciously) malformed. To ensure security in this setting, a party must run $\mathsf{VfyCK}(ck)$ before it ever commits, and if verification fails, it must not use the commitment to commit values or accept commitment openings. Consequently, the definitions of binding and hiding are now modified, and ck is provided by the responsible party (which is potentially malicious). Moreover, the in both the binding and the hiding game, the adversary loses if $\mathsf{VfyCK}(ck) = 0$.

Remark C.2.2 (Remark C.2.1 continued). The graph 3-colouring protocol $\mathrm{G3C}_{\mathrm{GK}}$ of Goldreich and Kahan [GK96] relies on a weaker "a posteriori hiding" property for a statistically hiding 3-move commitment scheme in the plain model. Here, VfyCK may depend on secrets, e.g. the randomness of Setup, allowing more candidates schemes. The verification secrets are only revealed after the binding property is not needed anymore. The win condition in the binding game is modified accordingly.

Concretely, in the zero-knowledge proof system from Goldreich and Kahan [GK96], the verifier commits to random challenges during the protocol. They challenge must be statistically hidden until it is unveiled. However, it suffices that the verifier is ensured of this statistical hiding property at the end of the protocol, since it can (and will) simply reject a purported proof the commitment key ck does not satisfy the statistical hiding property. Thus, delaying VfyCK and using the random coins for Setup in VfyCK makes sense in this special case.

C.2.2. Security Against Other Polynomial Runtime Classes

Security notions, i.e. binding and hiding, of commitments was only defined for a priori PPT adversaries in Section 2.2.4. To adapt these notions to more other polynomial runtime classes, let $\mathcal{T} \in \{\mathcal{PPT}, \mathcal{EPT}, \mathcal{CPPT}, \mathcal{CEPT}\}$. Observe that the definition of advantage of an adversary $\mathcal{A}$ against the binding or hiding property (Definitions 2.2.9 and 2.2.11) of a commitment scheme COM does not depend on $\mathcal{A}$'s runtime class. Thus, it is indeed straightforward to define binding and hiding for $\mathcal{T}$-time adversaries, as follows: COM is hiding (resp. binding) against $\mathcal{T}$-time adversaries, if for every $\mathcal{T}$-time adversary $\mathcal{A}$ the advantage of $\mathcal{A}$ is negligible. Note that we do not need to decide whether a $\mathcal{T}$-time adversary is defined by using the runtime of $\mathcal{A}$ only, or by using runtime of the whole experiment (including the challenger), because all algorithms of COM are a priori PPT anyway.

Finally, we note that CEPT adversaries are "no better" than a priori PPT adversaries.

Lemma C.2.3. *Suppose* COM *is computationally (resp. statistically, resp. perfectly) hiding (resp. binding) against a priori PPT adversaries. Then it also is against CEPT adversaries.*

Proof sketch. Use that COM consists of a priori PPT algorithms and a standard truncation to a priori PPT to obtain an adversary $\mathcal{A}'$ with advantage at least half the advantage of $\mathcal{A}$ infinitely often. □

C.3. ⋆ Technical lemmata

In this section, we gather some lemmata for various purposes. Appendix C.3.2 contain some simple facts on statistical distance. In Appendix C.3.3, some cryptographic results concerning distinguishing and general hybrid arguments are given. And Appendix C.3.4 contains naive closeness tests.

C.3.1. Tail Bounds

Tail bounds for distributions are the core tool for (runtime) cutoffs. For example, they allow to estimate how much the adversarial advantage suffers if we truncate.

Definition C.3.1 (Tail bounds). Let X be some distribution on $\mathbb{R}_{\geq 0}$. We call a (right-)continuous decreasing function $\mathsf{tail}\colon \mathbb{R}_{\geq 0} \longrightarrow \mathbb{R}_{\geq 0}$ a **tail bound of** X if $\forall x \in \mathbb{R}_{\geq 0}\colon \Pr[X > n] \leq \mathsf{tail}(n)$.

Moreover, we write $\mathsf{tail}^{\dagger}\colon \mathbb{R}_{\geq 0} \longrightarrow \mathbb{R}_{\geq 0} \cup \{\infty\}$ for $\mathsf{tail}^{\dagger}(\alpha) = \inf\{n \mid \mathsf{tail}(n) \leq \alpha\}$, which satisfies $\mathsf{tail}(\mathsf{tail}^{\dagger}(\alpha)) \leq \alpha$. More generally, we call an upper bound bnd of some sequence $(x_n)_n$ a tail bound, i.e. $x_n \leq \mathsf{bnd}(n)$ for all n.

Tail bounds generalize to distributions over $\mathbb{R}_{\geq 0} \cup \{\infty, \mathtt{timeout}\}$, etc.

The optimal tail bound is $\mathsf{tail}(n) = 1 - \mathsf{CDF}_X(n)$, where CDF_X is the cumulative distribution function of X. We use $\mathsf{tail}^{\dagger}(\alpha)$ to conveniently denote the minimal n_α with $\mathsf{tail}(n_\alpha) \leq \alpha$, which exists due to continuity of tail.

For *strict* runtimes, e.g. strict polynomial time, the time bound is an admissible tail bound. More generally, we recall following lemma:

Lemma C.3.2 (Markov bound). *Let X be a distribution on $\mathbb{R}_0$ and suppose $\mathbb{E}[X] \leq t$. Then $\mathsf{tail}(n) = \frac{1}{n}t$ is an admissible tail bound and $\mathsf{tail}^{\dagger}(\alpha) = \frac{1}{\alpha}t$. For $\|X\|_p = (\mathbb{E}[X^p])^{1/p} \leq t$ with $p \geq 1$, we have $\mathsf{tail}(n) = (\frac{t}{n})^p$, and hence $\mathsf{tail}(n) \leq \frac{t}{n}$ if $n \geq t$.*

For simple corollaries concerning runtime truncation and bounds, see Appendix C.3.3.

C.3.2. Simple Facts

In this section, we state some simple facts. Most are used with, or about, random variables, conditional variables, and the behaviour of statistical distance.

C.3.2.1. Statistical Distance

Following lemma is useful to bound statistical distances of products of densities.

Lemma C.3.3. *Let $p_i, q_i \in [0,1]$ for $i = 1, \ldots, n$. Then*

$$\left|\prod_{i=1}^{n} p_i - \prod_{i=1}^{n} q_i\right| \leq \sum_{i=1}^{n} |p_i - q_i|.$$

In particular, $\Delta(X \times Y, X' \times Y') \leq \Delta(X, Y) + \Delta(X', Y')$ for random variables X, Y, X', Y' (not necessarily independent).

More precisely, let $p_{(1,\ldots,k)} := \prod_{i=1}^{k} p_i$, and let $q_{(k,\ldots)} := \prod_{i=k}^{n} q_i$, and let $\delta_i := |p_i - q_i|$ then

$$\left|\prod_{i=1}^{n} p_i - \prod_{i=1}^{n} q_i\right| \leq \sum_{i=1}^{n} p_{(1,\ldots,i-1)} \delta_i q_{(i+1,\ldots)}.$$

Assuming the products are finite, this continues to hold for $n = \infty$.

Proof. This follows from a straightforward induction (using $|p_i|, |q_i| \leq 1$) to simplify. The claim regarding statistical distance follows by an application of the inequality under the integral. □

Next, we note how conditional distributions and statistical distance are connected.

Remark C.3.4. Let X be random variable and let Y independently distributed like X conditioned on some event of probability ε. Then $\Delta(X, Y) = \varepsilon$.

(This follows easily since Y has the density $\Pr[\mathcal{E}]^{-1}\mathbb{1}_{\mathcal{E}}$ as density w.r.t. X, where $\mathbb{1}_{\mathcal{E}}$ hence $2\,\Delta(X, Y) = \|\mathbb{1} - \Pr[\mathcal{E}]^{-1}\mathbb{1}_{\mathcal{E}}\|_1 = \Pr[\mathcal{E}] + \Pr[\mathcal{E}] = 2\varepsilon$.)

Following is a simple result of CDFs.

Corollary C.3.5. *Let X and Y be two random variables over $\mathbb{N}_0 \cup \{\infty\}$ and let $N \in \mathbb{N}_0$. Suppose X (resp. Y) are truncated to $X^{\leq N}$ (resp. $Y^{\leq N}$) (i.e. they output* `timeout` *if they exceed N). Then*

$$\Delta(X, Y) - \Pr[X > N] \leq \Delta(X^{\leq N}, Y^{\leq N}) \leq \Delta(X, Y).$$

Proof. We show $\Delta(X, Y) - \Delta(X^{\leq N}, Y^{\leq N}) \geq \Pr[X > N]$. The left-hand side is $\sum_{k=n}^{\infty} |p_X(k) - p_Y(k)| - |\sum_{k=n}^{\infty} p_X(k) - p_Y(k)|$. This can be interpreted as ℓ_1-norms and the claim follows by general inequalities, see Lemma C.3.6. □

Lemma C.3.6. *Let x, y be be two elements in a normed vector space and suppose $\|y\| \leq \varepsilon$. Then*

$$|\|x - y\| - |\|x\| - \|y\|\|| \leq 2\|y\| \leq 2\varepsilon$$

The inequality is tight ($y = -x$).

Proof. We consider two cases. Suppose $\|x\| \leq \|y\|$. Then we find

$$\|x - y\| - |\|x\| - \|y\|| = \|x - y\| - \|x\| + \|y\| \leq \|x - y - x\| + \|y\| = 2\|y\|.$$

For the case $\|y\| \leq \|x\|$ we find by symmetry (of $|a - b|$) that

$$\|y - x\| - |\|y\| - \|x\|| \leq 2\|y\| \leq 2\varepsilon.$$

This finishes the proof. □

C.3.2.2. Domination (with Slack)

We give some simple properties of domination (with slack).

Lemma C.3.7 (Properties of domination). *Let $\mathbb{R}' = \mathbb{R} \cup \{\infty, \texttt{timeout}\}$. Let $X, Y \colon \Omega \to \mathbb{R}'$ be random variables and suppose $X \overset{d}{\leq}_L Y$ for $L \geq 1$. Then:*

1. *For any monotonely increasing continuous function $f \colon \mathbb{R}' \to \mathbb{R}'$, we have $f(X) \overset{d}{\leq}_L f(Y)$.*
2. *In particular, for $X, Y \colon \Omega \to \mathbb{R}'$, we have $X^{\leq t} \overset{d}{\leq}_L Y^{\leq t}$.*
3. *For any $\nu \in [0,1]$, we have $X^{\nu} \overset{d}{\leq}_L Y^{\nu}$. Moreover, even conditioned on $\neg\texttt{timeout}$, the respective X', Y' satisfy $X' \overset{d}{\leq}_L Y'$*
4. *If $X, Y \geq 0$, $\|X\|_p \leq L\|Y\|_p$ for $p \in [1, \infty]$.*
5. *For $i = 1, \ldots, n$ let $X_i, Y_i \colon \Omega \to \mathbb{R}'$ with $X_i \overset{d}{\leq}_L Y_i$ and $\lambda_i > 1$ with $\sum_{i=1}^n \lambda_i = 1$. Then $\sum_{i=1}^n X_i \overset{d}{\leq}_M \sum_{i=1}^n \lambda_i^{-1} Y_i$. In particular, $\sum_{i=1}^n X_i \overset{d}{\leq}_L n \sum_{i=1}^n Y_i$.*

Proof. Let f be as claimed. Let g_+ be defined as $g_+(y) = \inf f^{-1}(\{z \mid y \leq z\})$ and let g_- be defined as $g_+(y) = \sup f^{-1}(\{z \mid z \leq y\}))$ Then $g_-(f(x)) \leq x \leq g_+(f(x))$ by definition, and $g_\pm$ are monotone. Since f is right-continuous, $f(x) \leq y \iff x \leq g_+(y)$. Consequently, for all $c \in \mathbb{R}'$

$$\Pr[f(X) > c] = 1 - \Pr[f(X) \leq c] = 1 - \Pr[X \leq g_+(c)] = \Pr[X > g_+(c)].$$

Now, we find for all $c \in \mathbb{R}'$

$$\Pr[f(X) > c] = \Pr[X > g_+(c)] \leq L \Pr[Y > g_+(c)] = L \Pr[f(Y) > c],$$

which proves the first claim. The second follows immediately by setting f appropriately. The last claim, directly follows in simple cases (namely if quantile-truncation coincides with truncation at some c). In general, we use that

$$\Pr[X > c] \leq L \Pr[Y > c] \iff \Pr[Y \leq c] \leq \frac{L-1}{L} \Pr[X > c]$$

and the definition of quantile cutoff

$$\Pr[X^{\nu} \leq c] = \max\{1, \Pr[X \leq c]\}.$$

Conditioning on $\neg\texttt{timeout}$ simply means using $\max\{1, \frac{1}{1-\nu} \Pr[X \leq c]\}$ as the new CDF. In both cases, the claim easily follows.

For the norm inequality, let $F = \mathrm{CDF}_X(\,\cdot\,)$, $G = \mathrm{CDF}_Y(\,\cdot\,)$ and note that $1 - F \leq L(1 - G)$ by assumption. Also recall that $\|X\|_1 = \mathbb{E}[X] = \int_0^\infty 1 - F(x) \mathrm{d}x$ for any distribution $X \geq 0$. We assume $p < \infty$, and leave $p = \infty$ to the reader. Thus, $\|X\|_p^p = \|X^p\|_1^p = \int_0^\infty 1 - F(x^{1/p}) \mathrm{d}x$. Finally $\int_0^\infty 1 - F(x^{1/p}) \mathrm{d}x \leq \int_0^\infty L(1 - G(x^{1/p})) \mathrm{d}x = L^p \|Y\|_p^p$.

For item 5, note that $\sum_i X_i > t$ implies that there exists some i such that $X_i > \lambda_i t$. Thus

$$\Pr[\sum_i X_i > t] \leq \sum_i \Pr[X_i > \lambda_i t] \leq \sum_i L \Pr[Y_i > \lambda_i t] \leq L \sum_i \Pr[\lambda_i^{-1} Y_i > t]$$

and the claim follows. □

C.3.3. Useful Lemmata

In this section, we give some simple lemmata, which are useful tools for moving back and forth between strict and expected time. The results given in this section are not asymptotic, and given for simple objects. Nevertheless, it is straightforward to show that all constructions can be directly applied in the asymptotic setting.

C.3.3.1. Runtime Truncations

We give generic variants of runtime truncation lemmata.

Corollary C.3.8. *Suppose* A *is some algorithm. Suppose* $\mathsf{A}(x)$ *takes an expected number of* t_x *steps on input* x*. Then the output distribution of* $\mathsf{A}(x)^{\leq N}$*, has statistical distance at most* $\frac{t_x}{N}$ *from* $\mathsf{A}(x)$*.*

Corollary C.3.8 bounds the quality loss when converting expected to strict time algorithms. For example, if A is a distinguisher with advantage ε, and $t_x \leq t$ for all inputs, then truncating runtime after $2\varepsilon^{-1}t$ steps yields a distinguisher with advantage $\frac{1}{2}\varepsilon$. If $t = \mathsf{poly}$ and $\varepsilon \geq 1/\mathsf{poly}$, then this transforms an *expected* polynomial time distinguisher into a *strict* polynomial time distinguisher.

Corollary C.3.9 (Non-asymptotic generic "standard reduction")**.** *Suppose* $\mathcal{D}^{\mathcal{O}}$ *is a distinguisher with advantage* ε *for* timed *oracles* $\mathcal{O}_0$*,* $\mathcal{O}_1$*. Let* $T_0 = \mathsf{time}_{\mathcal{D}+\mathcal{O}}(\mathcal{D}^{\mathcal{O}_0})$*, and let* $N = \mathsf{tail}^{\dagger}_{T_0}(\frac{\varepsilon}{4})$*. Then there is an* $\mathcal{A}$ *with runtime* $S_b = \mathsf{time}_{\mathcal{D}+\mathcal{O}}(\mathcal{A}^{\mathcal{O}_b})$ *for* $b = 0, 1$ *bounded roughly by* N *(plus overhead for computing* N *and emulating* $\mathcal{D}$*), and* $\mathcal{A}$ *distinguishes* $\mathcal{O}_0$ *and* $\mathcal{O}_1$ *with advantage* $\frac{\varepsilon}{4}$*.*

More precisely, $\mathcal{A}$ *truncates the total time of* $\mathcal{D} + \mathcal{O}$ *to at most* N *steps, hence the runtime distribution of* $\mathcal{A}$ *is close to that of* $\mathcal{D} + \mathcal{O}$*. Moreover, there are two possible candidates for* $\mathcal{A}$*: One outputs the output of* $\mathcal{D}$*, and a random guess in case of* `timeout`*. The other outputs* 1 *in case of* `timeout` *and* 0 *else. At least one of these algorithms has advantage* $\frac{\varepsilon}{4}$*.*

We note again, that only the runtime with $\mathcal{O}_0$ and its tail bound are of importance for the runtime cutoff. Also, one can trade-off runtime for advantage, e.g. by truncating at $N = \mathsf{tail}^{\dagger}_{T_0}(\frac{\varepsilon}{\mathsf{poly}})$. This cutoff argument and its variations play the role of the standard reduction to PPT (Corollary 5.4.2) in the general setting. We point out, that runtime is not the only (complexity) measure of interest which can be used in Corollary C.3.9. Besides elapsed runtime of $\mathcal{D} + \mathcal{O}$, the elapsed runtime of only $\mathcal{D}$, consumed memory, number of queries, query length, etc., are possible measures to which Corollary C.3.9 generalizes straightforwardly.

Proof sketch. Distinguisher $\mathcal{A}$ emulates $\mathcal{D}$ and truncates $\mathcal{D}$'s and $\mathcal{O}$'s combined steps to N. That is, $\mathcal{A}$ keeps track of the steps $t_{\mathcal{D}}$ and $t_{\mathcal{O}}$ and relies on $\mathcal{O}$ being a *timed* oracle to allow it a time bound of $N - t_{\mathcal{O}} - t_{\mathcal{D}}$ when invoked. Note that $\mathcal{A}$ emulates an priori number bounded number of N steps. Truncating $\mathcal{D}^{\mathcal{O}_0}$ after N steps w.r.t. oracle-included steps ensures that the output of $\mathcal{D}^{\mathcal{O}_0}$ has statistical distance at most $\frac{\varepsilon}{4}$.

Suppose the output of $\mathcal{A}^{\mathcal{O}_1}$ has statistical distance δ of $\mathcal{D}^{\mathcal{O}_1}$. If $\delta \geq \frac{2\varepsilon}{4}$. then necessarily, the probability that $T_1 = \mathsf{time}_{\mathcal{D}+\mathcal{O}}(\mathcal{D}^{\mathcal{O}_1})$ exceeds N steps is larger than $\frac{2\varepsilon}{4}$. Thus, this runtime statistic can be used as a distinguishing property, with advantage at least $\frac{\varepsilon}{4}$ infinitely often. (Concretely, $\mathcal{A}$ returns 1 if N steps are exceeded and 0 otherwise.)

Now suppose the probability that $T_1 = \mathsf{time}_{\mathcal{D}+\mathcal{O}}(\mathcal{D}^{\mathcal{O}_1})$ exceeds N steps is less than $\frac{2\varepsilon}{4}$. Let $\mathcal{A}$ guesses randomly in case of `timeout`. Then possible loss in advantage is bounded by $\frac{\varepsilon}{4} + \frac{2\varepsilon}{4} = \frac{3\varepsilon}{4}$. This leaves an advantage of $\frac{\varepsilon}{4}$ and the claim follows. □

Importantly, the construction of the two distinguisher candidates is uniform, and translates to the asymptotic setting. One of them has infinitely often advantage at least $\frac{\varepsilon}{4}$.

C.3.3.2. Hybrid Lemmata

Hybrid arguments and therefore the hybrid lemma are omnipresent in cryptography. Unfortunately, the standard hybrid lemma for strict polynomial time does not hold without change.

Example C.3.10 (Expected polynomial rounds). The need to deal with a priori infinitely many hybrids arises naturally from expected polynomial interaction: We have $\sum_{i\geq 1} 2^{-i} = 1$, so repeating some protocol (step) with probability $\frac{1}{2}$ implies an expected constant number of repetitions. But replacing each call by a simulation requires an infinite number of hybrid steps. Evidently, after replacing the first λ protocols by simulations, the remainder can be replaced in a single step, because more than λ repetitions are necessary only with probability $2^{-\lambda}$.

We state in general the truncation approach from Example C.3.10.

Corollary C.3.11 (Hybrid lemma). *Let $\mathcal{O}^0, \mathcal{O}^1, \dots, \mathcal{O}^\infty$ be oracles. Let $\mathcal{Z}_0$, $\mathcal{Z}_1$ be two more oracles, and let Z be an algorithm as follows: Z takes as input an integer $i \in \mathbb{N}$. Moreover, $\mathsf{Z}(i, \mathcal{Z}_b)$ is implements an oracle which behaves exactly like $\mathcal{O}^{i+b}$.*

Let $\mathcal{D}$ be a distinguisher for $\mathcal{O}^0$ and $\mathcal{O}^1$ with advantage ε, that is

$$|\Pr[\mathcal{D}^{\mathcal{O}^0} = 1] - \Pr[\mathcal{D}^{\mathcal{O}^\infty} = 1]| \geq \varepsilon$$

and suppose that we have a (tail) bound bnd *with*

$$|\Pr[\mathcal{D}^{\mathcal{O}^i} = 1] - \Pr[\mathcal{D}^{\mathcal{O}^\infty} = 1]| \leq \mathsf{bnd}(i).$$

Then for every $\alpha \leq \varepsilon$ there is a distinguisher $\mathcal{D}'$ which distinguishes $\mathcal{Z}_0$ and $\mathcal{Z}_1$ with advantage[11]

$$\varepsilon' = \frac{\varepsilon - \alpha}{N_\alpha} \quad \text{where } N_\alpha := \mathsf{bnd}^\dagger(\alpha).$$

More concretely, $\mathcal{D}'$ picks a random $i \overset{\$}{\leftarrow} \{0, N_\alpha - 1\}$, runs $\mathcal{D}$ on $\mathsf{Z}(i, \mathcal{Z}_b)$ and returns $\mathcal{D}$'s guess bit as its own. Thus, the runtime distribution of $\mathcal{D}'$ is closely related to that of $\mathcal{D}$ and Z.

Proof of Corollary C.3.11. We reduce the proof to the standard hybrid lemma. Note that it suffices to apply the standard hybrid lemma (with a finite number of steps) to $\mathcal{O}^0, \dots, \mathcal{O}^{N_\alpha}$. Because, by the very definition N_α we know that $|\Pr[\mathcal{D}^{\mathcal{O}^0} = 1] - \Pr[\mathcal{D}^{\mathcal{O}^{N_\alpha}} = 1]| \geq \varepsilon - \alpha = \varepsilon'$. Now, the standard hybrid lemma yields our distinguisher and advantage. □

[11] We note that $N_\alpha = \infty$ is possible, in which case $\varepsilon' = 0$.

Our statement of the hybrid lemma differs from the standard one in minor points.[12] It allows an a priori infinite number of hybrids. And it postulates a bound bnd on the closeness of the i-th and final hybrid. Typically bnd bounds the statistical distance of the i-th and final hybrid and is derived as a tail bound, e.g. Markov bound (Lemma C.3.2) on runtime or number of oracle queries.

While one may hope for an "expected number of hybrids" loss, this is impossible in general, since an adversary could focus its advantage on the "tail hybrids". Any black-box-like reduction is unlikely to achieve better bounds.

Example C.3.12 (Optimality of the (truncated) hybrid argument). Consider following (non-adaptive) distinguishing game: The adversary sends a number n to the challenger. The challenger prepares n truly random r_i or n pseudo-random $r_i = \mathrm{PRG}(s_i)$, and the adversary must distinguish. Consider an adversary with distribution N of n, so that $\mathbb{E}[N] \le 3$. The hope, that the hybrid argument may only lose a factor of 3 in advantage, is false. One the one hand, it may be that all the advantage of an adversary is in the tail of the distribution. Without a (non-obvious) way to reach the (distribution of) state, there is no other way than to run the adversary long enough. This affects any black-box reductions. A pathological adversary may furthermore distribute its advantage evenly over the hybrids as well, e.g. by first picking the number q of queries, and then breaking the i-th embedding with probability $\frac{1}{q}$. Consequently, improving on the hybrid lemma seems close to impossible. Better reductions have to make use of more information.

As is well known and for completeness demonstrated in Example C.3.12, the tails of distributions are a limiting factor for (hybrid) reductions. Nevertheless, Corollary C.3.11 is useful and generally good enough, though it may have poor tightness properties.

C.3.4. Testing Closeness of Distributions

Given two distributions, we need a way to efficiently test how close they are. Again, we give a non-asymptotic lemma. But we note that in the cryptographic setting, we will tell apart (families of) distributions which are statistically far (in the asymptotic sense).

Problem C.3.13 (Closeness promise problem). Let X, Y be distributions (typically on $\{1, \dots, n\}$). The **closeness promise problem** (with parameter $\varepsilon > 0$) is the following: Decide whether $X \overset{d}{\equiv} Y$ or $\Delta(X, Y) > \varepsilon$. A tester A is an algorithm which, given sample (oracle) access to X and Y outputs a verdict (i.e. a bit) whether $X = Y$ or not. The error of a tester is (at most) δ, if

$$\Pr[\mathcal{D}^{X,X'} = \mathtt{same}] \ge 1 - \delta \quad \text{and} \quad \Pr[\mathcal{D}^{X,Y} = \mathtt{different}] \ge 1 - \delta$$

We speak of *testing* instead of *distinguishing* since it is a slightly stronger notion. A distinguisher may guess randomly if $X \overset{d}{\equiv} Y$, but always decide $X \neq Y$ correctly, but a tester may not. In particular, a tester with error δ has distinguishing advantage $1 - 2\delta$.

[12] Sometimes, the hybrid lemma is stated in a weaker form, merely ensuring the existence of an index i where distinguishing hybrids i and $i + 1$ has advantage $\ge \varepsilon/m$. This does not naively extended to the asymptotic setting. Assuming that for all i, $\mathcal{O}^i \overset{c}{\approx} \mathcal{O}^{i+1}$ (asymptotically) *does not* imply $\mathcal{O}^0 \overset{c}{\approx} \mathcal{O}^\infty$ (asymptotically). Trivial counterexamples exist. Hence, the reduction to a (single) fixed indistinguishability assumption is essential for asymptotic usage of Corollary C.3.11.

Lemma C.3.14. *Let X, Y be distributions with support contained in on $\{1, \ldots, m\}$ and consider the closeness promise problem. Let $\varepsilon, \delta \in (0, 1]$. Then there is an algorithm* A *which solves the closeness promise problem with error δ and requires*

$$N = \lceil 6m\varepsilon^{-2} \log(2\delta^{-1}) \rceil$$

samples (of both X and Y). Moreover, A *is makes a linear number of arithmetic operations (in N).*

The result generalizes to any X, Y with support contained in $\mathcal{S}$, where $\operatorname{card}(\mathcal{S}) = m$*, since only comparison operations for elements are needed.*

We note that better closeness testing algorithms are known, namely in [CDVV14] an *optimal* closeness tester is given. That tester has linear runtime in the number of samples N as well.

Proof of Lemma C.3.14. Our tester simply uses the Kolmogorov–Smirnov test. That is, compute the empirical CDF F_X and F_Y (with N samples each) and test whether $\|F_X - F_Y\|_\infty < \varepsilon$. By applying a Chernoff bound argument in case $X \overset{d}{\not\equiv} Y$, and using the sharp Dvoretzky–Kiefer–Wolfowitz inequality by Massart in case $X \overset{d}{\equiv} Y$, we arrive at the claimed result. (Our constants are chosen so that we obtain $(\varepsilon/3, \delta/2)$ approximations of the true CDF's. By a standard argument using the triangle inequality, one obtains our claims.) □

As with the hybrid lemma, we have to deal with distributions with infinite support. Using tail bounds, we stretch Lemma C.3.14 to this case.

Corollary C.3.15. *Let X, Y be distributions on $\mathbb{N}_0$ and consider the closeness promise problem. Let $\varepsilon, \delta \in (0, 1]$ and let $\mathsf{tail}_X(\cdot)$ be a tail bound for X. Suppose $\varepsilon' = \varepsilon - \alpha$, where $\alpha > 0$, let $m' = \mathsf{tail}_X^\dagger(\alpha)$. Then there is an algorithm* A *which solves the closeness promise problem with error δ and requires*

$$N' = \lceil 6(m' + 1)\varepsilon'^{-2} \log(2\delta^{-1}) \rceil$$

samples (of both X and Y). Moreover, A *is only requires a linear number of arithmetic operations (in N').*

We note that $\mathbb{N}_0 \cup \{\infty\}$ (and the like) are also domains for which Corollary C.3.15 holds. It should also be evident, that this generalizes, as long as we can approximate X and Y suitably precise over a suitably small domain. Hence tail bounds are just a special case, and replacing X, Y by suitable close X', Y' works as well.

Proof. The algorithm simply maps the distributions X, Y to new distributions by mapping any sample s to $\max\{s, m\}$.[13] This changes the statistical distance by at most $\mathsf{tail}_X(m)$, see Corollary C.3.5. Now, apply Lemma C.3.14. □

Following remark, while a triviality, points out one core tool of this work.

[13] Note that this mapping does not need to "read" all of s (given e.g. tape-access starting from the least significant bit). In particular, in suitable machine models, we do not run into problems where some values s are gigantic and could not be read without compromising efficiency.

Important Remark C.3.16 (Statistical and computational indistinguishability coincide for "small" support). From Lemma C.3.14 and Corollary C.3.15, we already observe the following: Asymptotically, any pair of (families of) distributions X, Y, where one, say X, has (essentially) polynomial sized support $\{0, \dots, \mathsf{poly}(\lambda)\}$ are *computationally* triple-oracle indistinguishable under repeated sampling in polynomial time, if and only if, they are *statistically* triple-oracle indistinguishable under repeated sampling.

Remark C.3.17. Merely considering the domain, independently of X is a very rough point of view. After all, X could be concentrated on a tiny subset of $\{0, \dots, n\}$. In particular, relying on $\mathrm{supp}(X) \subseteq \mathbb{N}_0$ and using a total ordering and tail bounds, is not at all necessary. We consider a more sensitive closeness testing lemma a useful tool for more precise analysis. But the coarse (non-optimal) results stated here are good enough for our purposes.

C.4. ★ General Runtime Definitions

This section is (only) for the inclined reader. It contains our "general" treatment of runtime classes, that is, our framework and the many definitions necessary to talk about runtime classes and their properties. Unfortunately, we fall short of going beyond algebra-tailed runtime classes, hence by and large, nothing of essence is covered that is not already visible for polynomial time, PPT, EPT and CEPT.

C.4.1. Preliminaries: Bound Algebras

Most of our arguments work for runtime classes related to bound algebras, for example, the algebra of polynomials.

Definition C.4.1 (Bound algebras). A **bound algebra** $\mathcal{B}$ is a subset of $\mathbb{R}_{\geq 0}^{\mathbb{N}_0}$, i.e. a subset of sequences in $\mathbb{R}_{\geq 0}$, which satisfies:

- $\mathcal{B}$ is the subset of non-negative sequences of a subalgebra of $\mathbb{R}^{\mathbb{N}_0}$. In particular, it is closed under multiplication and it contains the constant 0 and constant 1 sequences.[14]
- $\mathcal{B}$ is closed under domination, i.e. $(x_\lambda)_\lambda \in \mathcal{B}$, then so is any $(y_\lambda)_\lambda$ with $y_\lambda \leq x_\lambda$ (for all λ).
- $\mathcal{B}$ is "asymptotically monotone": If $(x_\lambda)_\lambda \in \mathcal{B}$, then so is $(y_\lambda)_\lambda$ with $y_\lambda \coloneqq \max_{i=1}^{\lambda} x_i$.

A subset $\mathcal{G} \subseteq \mathcal{B}$ is *generates* $\mathcal{B}$ if for any $(x_\lambda) \in \mathcal{B}$ there is a $(y_\lambda) \in \mathcal{G}$ with $(x_\lambda) \leq (y_\lambda)$. The set $\mathsf{Negl}_{\mathcal{B}}$ of $\mathcal{B}$-**negligible functions**, is defined as $\mathsf{Negl}_{\mathcal{B}} = \{f \mid \limsup_{\lambda\to\infty} |f(\lambda)\mathsf{bnd}(\lambda)| = 0\}$.

When we work with bounds we often implicitly assume they are monotone.

Example C.4.2. Suitable function algebras, e.g. polynomials, or polylogarithmic functions, or $f(\lambda) = n^{\mathsf{polylog}(\lambda)}$, etc., induce a bound algebra. Importantly, there typically are monotone generating subsets (of countable size), e.g. $\{(c\lambda^c) \mid c \in \mathbb{N}_0\}$, which generate $\mathcal{B}$.

[14] The associated subalgebra of $\mathcal{B}$ is unique.

C.4.2. Runtime Distributions

Our definitions of (polynomial) runtime are such that an algorithm's (or protocol's) runtime is bounded *in the security parameter λ alone*. The input space of an algorithm is (a family) $\mathcal{X}_\lambda$.[15] Often, our algorithms have no (explicit) input, but receive implicit input via oracles, e.g. when distinguishing distributions given sampling access. In any case, we focus on "a posteriori" runtime, i.e. consider runtime $\mathsf{time}_{\mathsf{A}}(\mathsf{A}(x))$ where $x \xleftarrow{\$} \mathcal{X}$ for some input *distribution* (that is $\mathsf{A}(\mathcal{X})$ is a system *without* inputs).

Caution C.4.3. Recall that we generally suppress mentioning dependencies on the security parameter, i.e. we typically write $\mathsf{A}(x)$ instead of $\mathsf{A}(\lambda, x)$ if λ. The security parameter is (implicit) "input" to every algorithm. In fact, usually, A is given no inputs (but λ). Similarly, runtime obviously depends on the machine model even though we do not mention this.

Definition C.4.4 (Runtime distribution). A *(input-free)* **runtime (distribution)** T is a family $(T_\lambda)_\lambda$ of distributions $T_\lambda \in \mathsf{Dists}(\mathbb{N}_0 \cup \{\infty\})$ parameterized by λ; more precisely, it is a map $T\colon \mathbb{N}_0 \to \mathsf{Dists}(\mathbb{N}_0 \cup \{\infty\})$ from security parameter to probability distributions over $\mathbb{N}_0 \cup \{\infty\}$. A runtime T is **induced** by an algorithm A if $T_\lambda = \mathsf{time}_{\mathsf{A}}(\mathsf{A}(\lambda))$. We typically suppress λ and simply write $T = \mathsf{time}_{\mathsf{A}}(\mathsf{A})$.

We allow the symbol $\mathtt{timeout}$ in a runtime distribution T (formally changing to $\mathsf{Dists}(\mathbb{N}_0 \cup \{\mathtt{timeout}\})$).[16]

Remark C.4.5. Runtime (distributions) with input, or *input-dependent* runtimes are functions mapping input $x \in \mathcal{X}_\lambda$ to a runtime distribution, that is $T_\lambda\colon \mathcal{X}_\lambda \to \mathsf{Dists}(\mathbb{N}_0 \cup \{\infty\})$ for all λ. It is **induced** by A if $T_\lambda(x) = \mathsf{time}(\mathsf{A}(\lambda, x))$. The definition of input-dependent runtime (as a random variable) is similar.

For now, we only consider the input-free setting, i.e. $\mathcal{X} = \{\star\}$. Input is implicitly made available via oracle access.

Caution C.4.6. In this and future sections, we conflate *runtimes* (random variables) *runtime distributions*. The reason is, that we almost always care only about the runtime distribution, except in cases where we "split" up the runtime of an algorithm into a sum of stochastically dependent runtimes (e.g. of A and $\mathcal{O}$).

C.4.3. Runtime Classes

To talk about "efficient" computation, we need to say which runtime distributions we consider "efficient". The set of all "efficient" runtimes then forms the respective runtime class. Exemplary runtime classes are $\mathcal{PPT}$ and $\mathcal{EPT}$. We refine Definition 5.2.4 here, to only include sets of runtimes which have some basic properties.

Definition C.4.7. A **runtime class** $\mathcal{T}$ is a set of *input-free* runtime distributions so that:

Constants: The constant 0 and constant 1 runtime are in $\mathcal{T}$.

Closed under domination: that is, if $T \in T$ and $S \leq T$ then $S \in \mathcal{T}$.[17]

[15] Recall a well-known problem: The input space may not be (efficiently) recognizable. Thus, an algorithm may be fed with malformed input (or oracles/interaction). In general, this voids any runtime guarantees. Thus, for protocols, we want strong runtime guarantees, which are not restricted do well-formed input.

[16] We could also allow ∞ there, but generally timeouts stop overlong executions.

[17] More precisely, $S \leq T$ iff for all λ we have $S_\lambda \leq T_\lambda$, i.e. T_λ dominates S_λ in distribution.

Closed under addition, i.e. $\mathcal{T} + \mathcal{T} \subseteq \mathcal{T}$, where $T + S$ is viewed as a sum of distributions.

An (oracle) algorithm A runs in $\mathcal{T}$**-time** if $\mathsf{time}(\mathsf{A}) \in \mathcal{T}$.

Closure under domination says that no “inefficient” algorithm (i.e. runtime outside $\mathcal{T}$) can be made efficient by doing *more* steps. Additive closure roughly ensures that independent execution of any constant number of efficient algorithms is efficient. The definition of runtime class is most likely incomplete. We just give enough properties so that our results hold. Sensible runtime classes should offer more guarantees, but we have not identified the “right” properties.

Example C.4.8. We give some exemplary polynomial runtime classes.

Strict polynomial time: The runtime class $\mathcal{PPT}$ contains (by definition) all runtimes T for which there exists a polynomial poly such that $T \leq \mathsf{poly}$.

Expected polynomial time: The runtime class $\mathcal{EPT}$ contains (by definition) all runtimes T for which there exists a polynomial poly such that $\mathbb{E}[T] \leq \mathsf{poly}$, i.e. $\mathbb{E}[T_\lambda] \leq \mathsf{poly}(\lambda)$ for all λ.

Polynomial $\|\cdot\|_q$-time: By polynomially bounding $\|T\|_q$ (for $q \in [1, \infty]$), we generalize both strict ($q = \infty$) and expected time ($q = 1$). For example $q = 2$ implies polynomially bounded variation (and expectation).

Quasi-linear time: If we require $T_\lambda \leq \lambda \cdot \mathsf{polylog}(\lambda)$ we obtain quasi-linear runtime. This class only satisfies weak composition properties, and is not covered by our results.

Now, we generalize polynomial time bounds to algebra bounds. For that, we need following definition.

Definition C.4.9. We say that a runtime class $\mathcal{T}$ is **weakly compatible** with a bound algebra $\mathcal{B}$, if for any $\mathsf{bnd}_0 \in \mathcal{B}$, there is a $\mathsf{bnd}_1 \in \mathcal{B}$ so that bnd_1 can be computed in $\mathcal{T}$-time. More concretely, $\mathsf{bnd}_1(\lambda)$ can be computed in time T_λ for $T \in \mathcal{T}$.

We call $\mathcal{T}$ **(strongly) compatible** with $\mathcal{B}$ if additionally strict $\mathcal{B}$-time (see Definition C.4.10 below) is contained in $\mathcal{T}$.

Compatibility ensures that $\mathcal{T}$ and $\mathcal{B}$ behave well in reduction arguments. (Strong) Compatibility is simpler to work with than weak compatibility, since for example $\mathcal{PPT}$ is weakly compatible with $\mathcal{B} = 2^{\mathcal{O}(\lambda)}$, but does evidently not contain all strict $\mathcal{B}$ algorithms.

Definition C.4.10 (Bound algebras and runtime classes). Instead of polynomials, some (suitable) algebra $\mathcal{B}$ may be used for time bounds, e.g. $n^{\mathsf{polylog}(n)}$, see Definition C.4.1. By definition, we always require that the defined runtime class $\mathcal{T}$ is *compatible* with the bound algebra $\mathcal{B}$.

Algebra-bounded $\|\cdot\|_q$-time: We write $\mathsf{RTC}_q(\mathcal{B})$ for the runtime class containing all runtimes T with $\|T_\lambda\|_q \leq \mathsf{bnd}(\lambda)$ for some $\mathsf{bnd} \in \mathcal{B}$.

Algebra-tailed time: We generalize algebra-bounded time as follows: A runtime class $\mathcal{T}$ is $\mathcal{B}$**-tailed**, for a bound algebra $\mathcal{B}$, if: For every $T \in \mathcal{T}$, for every $\mathsf{bnd}_{\mathsf{tail}} \in \mathcal{B}$, there is a $\mathsf{bnd}_T \in \mathcal{B}$, such that $\Pr[T_\lambda > \mathsf{bnd}_T(\lambda)] \leq \frac{1}{\mathsf{bnd}_{\mathsf{tail}}(\lambda)}$ for all λ.[18]

We also refer to algebra-bounded times via *strict (or expected) $\mathcal{B}$-time.*

By Lemma C.3.2, any algebra-bounded runtime class is also algebra-tailed. Namely pick $\mathsf{bnd}_T = t \cdot \mathsf{bnd}_{\mathsf{tail}} \geq \mathsf{tail}^{\dagger}(\frac{1}{\mathsf{bnd}_{\mathsf{tail}}})$, where $t = \|T\|_q$. Also, Levin’s relaxation of EPT is polynomially-tailed.

[18] Recall that asymptotics, should be part of $\mathcal{B}$, so we use for *all* λ (and not for almost all).

We will focus on algebra-tailed runtime classes and runtime classes we derived from them.

Lastly, we define "abstract" runtime cutoffs.

Definition C.4.11 (Runtime truncation). Let T be a runtime. We define the **runtime cutoff** or **runtime truncation** $T^{\leq N}$ of T after N steps as the distribution (or random variable) given by $T|_{(\,\cdot\,>N)\mapsto\texttt{timeout}}$, i.e. by mapping any $k > N$ to $\texttt{timeout}$ (and the identity mapping otherwise). Runtime truncation is assumed to be an *efficient* oracle-transformation in any suitable machine model.[19]

Remark C.4.12. We stress that an efficient implementation of runtime cutoffs is vital for any results making use of them. We also note that this means that the *truncation bounds themselves must be efficiently computable.* This is ensured by the compatibility requirement in Definition C.4.10.

C.4.4. $\mathcal{T}$-time Triple-Oracle Indistinguishability

There are several notions of indistinguishability of distributions X_0, X_1 w.r.t. to $\mathcal{T}$-time algorithms. We choose indistinguishability *under repeated sampling* with *additional sampling access* to X_0 and X_1. The decision to give oracle sampling access the distributions X_0 and X_1, as well as the challenge distribution $Z \overset{d}{\equiv} X_b$ mirrors the fact that an algorithm can be (independently) executed many times, and should still remain efficient.[20] In particular, if $X_0 = \mathsf{time}(\mathsf{A}_0)$ is the runtime distribution of an efficient algorithm, and $X_1 = \mathsf{time}(\mathsf{A}_1)$ is inefficient, then X_1 is not efficiently samplable by emulating A_1. To simplify, we assume sampling access to both X_0 and X_1. Recall that we assume access to *binary* encodings of runtime.[21]

Another simplification is that we require *constant* distinguishing advantage. By standard amplification techniques, this is equivalent to non-$\mathcal{B}$-negligible success for algebra-tailed runtime classes.

Definition C.4.13 (Triple-oracle distinguishing). Let $\mathcal{O}_0$ and $\mathcal{O}_1$ be sampling oracles for distributions X_0, X_1 (i.e. oracles which return a fresh sample distributed as X_b when queried). Consider the distinguishing experiment $\mathrm{Exp}^{\text{3-dist}}_{\mathcal{A},\mathcal{O}_0,\mathcal{O}_1}$.

```
**Experiment** $\mathrm{Exp}^{\text{3-dist}}_{\mathcal{A},\mathcal{O}_0,\mathcal{O}_1}(\lambda)$
    $b \xleftarrow{\$} \{0,1\}$
    Instantiate an independent $\mathcal{O}^* := \mathcal{O}_b$
    $b' \leftarrow \mathcal{A}^{\mathcal{O}_0,\mathcal{O}_1,\mathcal{O}^*}(\lambda)$
    **return** $b' \overset{?}{=} b$
```

The distinguishing advantage of an algorithm $\mathcal{D}$ is defined as

$$\begin{aligned}\mathrm{Adv}^{\text{3-dist}}_{\mathcal{D},\mathcal{O}_0,\mathcal{O}_1}(\lambda) :=& 2\Pr[\mathrm{Exp}^{\text{3-dist}}_{\mathcal{A},\mathcal{O}_0,\mathcal{O}_1}(\lambda) = 1]\\ =& |\Pr[\mathcal{D}^{\mathcal{O}_0,\mathcal{O}_1,\mathcal{O}_1^*}(\lambda) = 1] - \Pr[\mathcal{D}^{\mathcal{O}_0,\mathcal{O}_1,\mathcal{O}_0^*}(\lambda) = 1]|,\end{aligned}$$

[19] This means that applying runtime cutoff to a *runtime oracle* is efficient. For example, given tape access to bit-encoded oracle results, we can read the minimal number of bits necessary to recognize $t > N$ and then return $\texttt{timeout}$.

[20] Our notion behaves nicely in almost any aspect, and agrees with standard notions if X_0 and X_1 are efficiently samplable (by a standard hybrid argument). We can amplify distinguishing advantage (as usual) and are guaranteed that *statistically indistinguishable distributions are statistically close.* Neither of this holds for the usual notions of one-sample or k-sample distinguishing, see for example [Mey94; GM98; GS98].

[21] A unary encoding would work as well, since we always reduce (a a priori strict time) distinguishers which use a strictly truncated version of the time. This truncated time can be read efficiently in both unary an binary.

where $\mathcal{O}_b^* = \mathcal{O}_b$, but independent. (The second equality only holds if $\mathcal{D}$ always returns a bit.) The randomness is taken over the algorithms and oracles randomness.

A distinguisher $\mathcal{D}$ is $\mathcal{T}$-time, if $\mathsf{time}_{\mathcal{D}}(\mathrm{Exp}^{\text{3-dist}}_{\mathcal{A},\mathcal{O}_0,\mathcal{O}_1}) \in \mathcal{T}$.[22] We call $\mathcal{O}_0$ and $\mathcal{O}_1$ **($\mathcal{T}$-time) computationally (triple-oracle) indistinguishable**, written $\mathcal{O}_0 \overset{c}{\approx}_{\mathcal{T}} \mathcal{O}_1$, if for all $\mathcal{T}$-time distinguishers $\mathcal{D}$,

$$\mathrm{Adv}^{\text{3-dist}}_{\mathcal{D},\mathcal{O}}(\lambda) \in o(1).$$

that is, any distinguisher has asymptotically vanishing advantage. Put differently, a computational distinguisher must have constant advantage $c > 0$ (for infinitely many λ). We define **$\mathcal{T}$-query statistical indistinguishability** as $\mathcal{T}$-time indistinguishability, where we only count a query to an oracle as a step (costing unit time).

We use Definition C.4.13 only for (runtime) distributions, and not general oracle-indistinguishability.

Remark C.4.14 (Why no general advantage classes?). For algebra-tailed runtime classes, using non-constant advantage, namely non-$\mathcal{B}$-negligible advantage, also works (due to amplification). We could define general "advantage classes", such as subexponentially negligible, polynomially negligible, or $1 - \frac{1}{\lambda}$. One reason not to do this is our focus on indistinguishability of runtimes, not in general distributions. We crucially rely on tail bounds and triple-oracle indistinguishability, which leads to (maybe unnecessary) limitation. In the "low advantage regime", e.g. subexponential advantage, it seems that the arguments based tail bounds do not carry over. In the "high advantage regime", e.g. advantage of at most $1 - 1/\mathrm{poly}$, the use of triple-oracle (in particular repeated samples) makes possible results uninteresting and useless (due to amplification to $1 - \exp(\lambda)$).

Alternative proof techniques, which do not rely on (repeated) sample access and tail bounds as their central tool are required. It is likely, that the approach(es) in Appendices C.5.3.1 and C.5.3.2 could be extended. This is out of scope for this work.

Since it is a useful point of view, we slightly generalize distinguishing. Namely, instead of directly outputting a verdict, one may output some processed information, which is fed into another distinguisher (perhaps repeatedly).

Remark C.4.15 (Generalized distinguisher). Let us call a distinguisher, which outputs not only 0 or 1, but different or additional information, a *generalized distinguisher.* Clearly, if two distributions are (computationally) indistinguishable, then the output of any generalized distinguisher is also (computationally) indistinguishable.

The upshot of this deliberation is that *any efficiently computable statistic* of an execution of a distinguisher $\mathcal{D}$ must be *indistinguishable.* Otherwise, there is a distinguisher $\mathcal{D}'$ which emulates $\mathcal{D}$ and uses that statistic to attack indistinguishability. In particular, *runtime* is such a statistic, and the number of oracle queries is another.

Now, we apply the notion of $\mathcal{T}$-time triple-oracle indistinguishability to runtimes.

Definition C.4.16. Suppose $\mathcal{T}$ is a (input-free) runtime class. Let T resp. S be (arbitrary) runtimes and suppose $\mathcal{O}_0$ resp. $\mathcal{O}_1$ sample T resp. S. We call T and S **(computationally) $\mathcal{T}$-time (triple-oracle) indistinguishable** if the respective distributions are (computationally) $\mathcal{T}$-time triple-oracle indistinguishable. We also write $T \overset{c}{\approx}_{\mathcal{T}} S$. The definition of **statistically $\mathcal{T}$-query (triple-oracle) indistinguishable** runtimes is analogous, written $T \overset{s}{\approx}_{\mathcal{T}} S$.

[22] Equivalently, $\mathsf{time}_{\mathcal{D}}(\mathcal{D}^{\mathcal{O}_0,\mathcal{O}_1,\mathcal{O}_b}) \in \mathcal{T}$ for $b = 0, 1$.

In the following, we always mean *triple-oracle* indistinguishable, if not otherwise specified. We come back to standard indistinguishability only in Appendix C.4.7

C.4.5. Closed Runtime Classes

Now, we come to a central definition, which applies the principle that $\mathcal{T}$-time indistinguishable objects should be considered "identical" for all cryptographic intents and purposes to $\mathcal{T}$-time itself.

Definition C.4.17 ($\mathcal{T}$-closed). Suppose $\mathcal{T}$ and $\mathcal{S}$ are runtime classes. We call $\mathcal{S}$ **computationally** (resp. **statistically**) $\mathcal{T}$**-closed** if following holds: For all runtimes S, if there is a runtime $\widetilde{S} \in \mathcal{T}$ and $S \overset{c}{\approx}_{\mathcal{S}} \widetilde{S}$ (resp. $S \overset{s}{\approx}_{\mathcal{S}} \widetilde{S}$), then $S \in \mathcal{S}$.

We call a runtime class $\mathcal{T}$ computationally (resp. statistically) **closed**, if it is $\mathcal{T}$-closed.

Example 5.1.5 demonstrates that neither $\mathcal{PPT}$ nor $\mathcal{EPT}$ is a closed runtime class. Before we define the closure of a runtime class, we give some helpful definitions.

Definition C.4.18 (Generating set). Let $\mathcal{U}$ be a *set* of runtimes. We say $\mathcal{U}$ **generates** $\mathcal{T}$ if $\mathcal{U} \subseteq \mathcal{T}$ and for any runtime class $\mathcal{T}'$ containing $\mathcal{U}$, we have $\mathcal{T} \subseteq \mathcal{T}'$. Equivalently, $T \in \mathcal{T} \iff \exists S \in \mathcal{U} \colon T \leq S$. Equivalently, $\mathcal{T}$ is the minimal runtime class containing $\mathcal{U}$.[23]

This shows that indistinguishability w.r.t. any generating subset $\mathcal{U} \subseteq \mathcal{T}$ or w.r.t. $\mathcal{T}$ coincides. For example, for $\mathcal{PPT}$, the set $\{\mathsf{poly}(\lambda) = n\lambda^n \mid n \in \mathbb{N}\}$ is generating, since every runtime is dominated by a runtime in this set.

Remark C.4.19. We can translate generating sets to the setting of bound algebras. Indeed, in Definition C.4.10, we require a generating set of efficiently computable bounds.

The perhaps most important relation between sets of runtimes is the following.

Definition C.4.20 (D-dense). A subset of runtimes $\mathcal{U} \subseteq \mathcal{T}$ is called **computationally** (resp. **statistically**) **distinguishing-dense** (short **d-dense**) in runtime class $\mathcal{T}$ if for any pair of distributions X, Y (over $\mathbb{N}_0 \cup \{\infty\}$) we have

$$X \overset{c/s}{\approx}_{\mathcal{T}} Y \implies X \overset{c/s}{\approx}_{\mathcal{U}} Y$$

w.r.t. triple-oracle indistinguishability. In other words, if $\mathcal{T}$ can distinguish two distributions, so can $\mathcal{U}$. A weakening of d-dense is **runtime d-dense**, where X must be in $\mathcal{T}$.

We note that d-density of $\mathcal{U} \subseteq \mathcal{T}$ is much different from being generating. For example, $\mathcal{PPT} \subseteq \mathcal{EPT}$ is d-dense, since any (successful) *expected* polynomial time distinguisher can be transformed into a (still successful) *strict* polynomial time distinguisher, see Corollary C.3.8.

Lemma C.4.21. *Let $\mathcal{T} \subseteq \mathcal{S}$ be runtime classes. Suppose that $\mathcal{S}$ is computationally $\mathcal{T}$-closed and that $\mathcal{T}$ is computationally (runtime) d-dense in $\mathcal{S}$. Then $\mathcal{S}$ is computationally closed. The same holds in the statistical case.*

[23] It is easy to see that an arbitrary intersection of runtime classes is again a runtime class. Hence, the generated runtime class of $\mathcal{U}$ is the intersection of all runtime classes containing $\mathcal{U}$, in particular, it exists and is unique.

Proof. Let $\widetilde{T} \in \mathcal{S}$ and let T be some runtime. Suppose $\widetilde{T} \overset{c}{\approx}_{\mathcal{S}} T$. Then $\widetilde{T} \overset{c}{\approx}_{\mathcal{T}} T$, since $\mathcal{T}$ is runtime d-dense in $\mathcal{S}$ and $\widetilde{T} \in \mathcal{S}$. Then $T \in \mathcal{S}$, since $\mathcal{S}$ is computationally $\mathcal{T}$-closed. The statistical case follows analogously. □

We now give a (constructive) definition of the closure of a runtime class.

Definition C.4.22 (Closure). Let $\mathcal{T}$ and $\mathcal{S}$ be a runtime classes. We define the computational $\mathcal{S}$-**closure** $\mathrm{Cls}^{c}_{\mathcal{S}}(\mathcal{T})$ of $\mathcal{T}$ as

$$\mathrm{Cls}^{c}_{\mathcal{S}}(\mathcal{T}) := \{S \colon \mathbb{N}_0 \to \mathsf{Dists}(\mathbb{N}_0 \cup \{\infty\}) \mid \exists T \in \mathcal{T} : S \overset{c}{\approx}_{\mathcal{S}} T\}.$$

The statistical $\mathcal{S}$-closure $\mathrm{Cls}^{S}_{\mathcal{S}}(\mathcal{T})$ is defined analogously. The **closure** $\overline{\mathcal{T}}$ of $\mathcal{T}$ is $\mathrm{Cls}^{c/s}_{\mathcal{T}}(\mathcal{T})$ (whether computational or statistical will be clear from the context).

An abstract notion of closure (e.g. minimal closed runtime class containing $\mathcal{T}$) and its equivalence with Definition C.4.22 would be a good justification for our definition.

Lemma C.4.23 (Closures are closed)**.** *The closure $\overline{\mathcal{T}}$ of a runtime class $\mathcal{T}$ is closed. (This holds in the computational and the statistical case.)*

Proof. Consider a runtime $T \in \overline{\mathcal{T}}$ and some arbitrary runtime S and suppose that $T \overset{c}{\approx}_{\overline{\mathcal{T}}} S$. To show that $\overline{\mathcal{T}}$ is closed, we need $S \in \overline{\mathcal{T}}$. Since $\mathcal{T} \subseteq \overline{\mathcal{T}}$, we have $T \overset{c}{\approx}_{\mathcal{T}} S$. By definition of $\overline{\mathcal{T}}$, there is some $\widetilde{T} \in \mathcal{T}$ such that $\widetilde{T} \overset{c}{\approx}_{\mathcal{T}} T$. Now, we have $\widetilde{T} \overset{c}{\approx}_{\mathcal{T}} T \overset{c}{\approx}_{\mathcal{T}} S$. This implies $S \in \overline{\mathcal{T}}$ by definition of $\overline{\mathcal{T}}$.[24] This proves the claim. The statistical case follows analogously. □

We would like a stronger result. We state this in following conjecture, which has little support for general runtime classes.

Conjecture C.4.24 (Closures are small). For any "benign" runtime class $\mathcal{T}$, $\mathcal{T}$ is runtime d-dense in $\overline{\mathcal{T}}$.

We expect that runtime classes where Conjecture C.4.24 fails behave rather strangely. While we do not know what "benign" runtime classes are or how to prove Conjecture C.4.24 in general, it is simple for algebra-tailed runtime classes.

Lemma C.4.25. *Let $\mathcal{B}$ be a bound algebra and $\mathcal{T}$ be $\mathcal{B}$-tailed. Then,* strict *$\mathcal{B}$-time is d-dense in $\overline{\mathcal{T}}$. (This holds in the computational and statistical case.)*

Proof sketch. Suppose $\mathcal{D}$ is a $\overline{\mathcal{T}}$-time distinguisher of distributions X and Y with advantage $\geq \varepsilon$ (for infinitely many λ and constant ε). Let $T = \mathsf{time}(\mathcal{D})$. We know that $T \overset{c}{\approx}_{\mathcal{T}} \widetilde{T}$ for some $\widetilde{T} \in \mathcal{T}$. Thus, for any $\mathcal{T}$-computable bound bnd, we have $|\Pr[T_\lambda \leq \mathsf{bnd}(\lambda)] - \Pr[\widetilde{T}_\lambda \leq \mathsf{bnd}(\lambda)]| \leq o(1)$. Otherwise T and $\widetilde{T}$ would be $\mathcal{T}$-time distinguishable.

Since $\mathcal{T}$ is $\mathcal{B}$-tailed, there exist an (efficiently computable) bound $\mathsf{bnd}(\lambda) \geq \mathsf{tail}^{\dagger}_{\widetilde{T}_\lambda}(\frac{2}{3}\varepsilon)$. Consequently, $\mathcal{D}^{\leq \mathsf{bnd}}$ is strict $\mathcal{B}$-time, hence $\mathcal{T}$-time, and retains a distinguishing advantage of $\frac{2}{3}\varepsilon - o(1)$ (infinitely often), which is at least $\frac{1}{2}\varepsilon$ infinitely often. □

[24] Triple-oracle indistinguishability is transitive for any *constant* number of hops.

We note an interesting step in the argument: The connection to $\mathcal{D}$'s runtime T is indirect, since we rely on $\widetilde{T}$ instead. We only needed suitable bounds for *truncation*. Indeed, runtime truncation seems to be the central (and only) tool at our disposal, and someway or another, it is what our proofs rely on.

Remark C.4.26 (Efficiency of truncations). Note that $\mathsf{time}_{\mathcal{D}}(\mathcal{D}^{\leq \mathsf{bnd}}) \leq \mathsf{time}_{\mathcal{D}}(\mathcal{D})$ (up to emulation overhead), that is, the truncation is "as efficient as" $\mathcal{D}$, and only loses advantage/output quality.

Remark C.4.27 (Non-negligible advantage). Lemma C.4.25 immediately extends to advantage $\varepsilon = 1/\mathsf{bnd}(\lambda)$ (for infinitely many λ). Just replace $o(1)$ by $\mathsf{negl}_{\mathcal{B}}$ and note that $\mathsf{tail}^{\dagger}_{\widetilde{T}_\lambda}(\alpha \frac{1}{\mathsf{bnd}(\lambda)}) \in \mathcal{B}$ for any constant $\alpha > 0$ and $\mathsf{bnd} \in \mathcal{B}$. This direct "conversion" to the usual setting of non-negligible advantage typically works for our results concerning algebra-tailed runtime classes.

Following lemma is useful to check if some runtime class $\mathcal{S}$ is the closure of $\mathcal{T}$.

Lemma C.4.28 (Closures are minimal). *Let $\mathcal{T} \subseteq \mathcal{S} \subseteq \overline{\mathcal{T}}$ be runtime classes. Suppose that $\mathcal{S}$ is $\mathcal{T}$-closed and $\mathcal{T}$ is d-dense in $\mathcal{S}$. Then $\mathcal{S} = \overline{\mathcal{T}}$. (This holds in the computational and statistical case.)*

Proof. Similar to Lemmas C.4.21 and C.4.23. (Any element in $\overline{\mathcal{T}}$ also lies $\mathcal{S}$.) □

Let us consider a simple concrete example.

Example C.4.29 (CPPT). We denote the closure of $\mathcal{PPT}$ as $\overline{\mathcal{PPT}}$ or $\mathcal{CPPT}$ and call it **computationally probabilistic polynomial time** (CPPT). In Appendix C.4.6, we find that statistical and computational closure coincide, hence "CPPT = SPPT". By definition, $\mathcal{CPPT}$ is

$$\mathcal{CPPT} = \{T \mid \exists \mathsf{poly}, \mathsf{negl}\colon \Pr[T_\lambda \geq \mathsf{poly}(\lambda)] \leq \mathsf{negl}(\lambda)\}.$$

In other words, CPPT relaxes PPT by allowing a negligible amount of superpolynomial executions. Now, we check that $\mathcal{CPPT} = \overline{\mathcal{PPT}}$. Clearly, $\mathcal{CPPT}$ contains $\mathcal{PPT}$. It is easy to see that, $\mathcal{PPT}$ is d-dense in $\mathcal{CPPT}$ and $\mathcal{CPPT}$ is $\mathcal{PPT}$-closed. Since also $\mathcal{CPPT} \subseteq \overline{\mathcal{PPT}}$, we find equality from Lemma C.4.21 and Lemma C.4.28.

C.4.6. Equivalence of Runtime-Indistinguishability for Algebra-Tailed Runtime Classes

In this section, we establish that for an algebra-tailed runtime class $\mathcal{T}$, statistical and computational $\mathcal{T}$-time indistinguishability coincide of runtime distributions. We give two such lemmata. The first one is simple and illustrates underlying reasons using strict algebra-bounded runtime classes. The second one extends this to algebra-tailed runtime classes. Both lemmata seem inherently limited to runtime classes containing a large enough "strict" subclass.

Lemma C.4.30. *Let $\mathcal{B}$ be a bound algebra and $\mathcal{T} = \mathsf{RTC}_\infty(\mathcal{B})$ be the corresponding* strict *runtime class. Let $T \in \mathcal{T}$ and let S be some runtime. Then $T \overset{s}{\approx}_{\mathcal{T}} S$ implies $T \overset{c}{\approx}_{\mathcal{T}} S$. More generally, if X and Y are distributions supported on a set S with cardinality $\mathsf{card}(S)$ in $\mathcal{B}$, then statistical and computational indistinguishability coincide. The (efficient) distinguisher is as in Lemma C.3.14 with parameters so that it runs in strict $\mathcal{B}$-time.*

Let X be a distribution or a random variable. For convenience, we write $X^{\{k\}}$ for the k-fold product distribution of *stochastically independent* products. That is, $(x_1, \dots, x_k) \xleftarrow{\$} X^{\{k\}}$ is distributed as $x_i \xleftarrow{\$} X$ for k independent samples x_i.

Proof. Note that computational distinguishability implies statistical distinguishability. To prove the converse, we invoke Lemma C.3.14. Let $\mathsf{bnd}_0 \in \mathcal{B}$ bound the support size of the distributions X, Y.[25] The key point is: If $X \overset{s}{\not\approx}_{\mathcal{T}} Y$, then the statistical distance is lower-bounded by $1/\mathsf{bnd}_{\mathrm{stat}}$ for some efficiently computable $\mathsf{bnd}_{\mathrm{stat}} \in \mathcal{B}$. Otherwise $\Delta(X^{\{\mathsf{bnd}\}}, Y^{\{\mathsf{bnd}\}}) \leq \mathsf{bnd} \cdot \Delta(X, Y) \in o(1)$ for all bnd, and hence $X^{\{\mathsf{bnd}\}}, Y^{\{\mathsf{bnd}\}}$ are statistically close, for any (statistical) distinguisher. A contradiction to triple-oracle distinguishability.

We invoke Lemma C.3.14 with $n = \mathsf{bnd}_0$, $\varepsilon = \frac{1}{2\mathsf{bnd}_{\mathrm{stat}}}$, and δ small enough, say $\delta = 1/8$. We obtain a distinguisher $\mathcal{D}$ with runtime roughly $24\mathsf{bnd}_0(\lambda)\mathsf{bnd}_{\mathrm{stat}}(\lambda)^2$ plus the overhead for evaluating $\mathsf{bnd}_0(\lambda), \mathsf{bnd}_{\mathrm{stat}}(\lambda)$. Thus, $\mathcal{D}$ is efficient. □

As we have seen, the equivalence between statistical and computational indistinguishability of runtimes follows because the support of a runtime distribution is "small", compared to the allotted runtime for distinguishers. This, of course, is by definition of runtime resp. "small".

Now, we generalize Lemma C.4.30 just like we generalized Lemma C.3.14 to Corollary C.3.15.

Corollary C.4.31. *Let $\mathcal{B}$ be a bound algebra and let $\mathcal{T}$ be a $\mathcal{B}$-tailed runtime class. Let X, Y be distributions over $\mathbb{N}_0 \cup \{\infty\}$ and suppose that X is $\mathcal{B}$-tailed, i.e. we have a tail bound* tail_X *such that*

$$\forall\, \mathsf{bnd} \in \mathcal{B}\colon\ \mathsf{tail}^{\dagger}_{X_\lambda}(\frac{1}{\mathsf{bnd}(\lambda)}) \in \mathcal{B}.$$

Then $X \overset{s}{\approx}_{\mathcal{T}} Y$ implies $X \overset{c}{\approx}_{\mathcal{T}} Y$. In particular, any runtime distribution $X = T \in \mathcal{T}$ is $\mathcal{B}$-tailed by assumption. The (efficient) distinguisher is as in Corollary C.3.15 with parameters so that it runs in strict *$\mathcal{B}$-time. In particular,* $\mathsf{RTC}_\infty(\mathcal{B})$ *is d-dense is in* $\mathsf{RTC}_q(\mathcal{B})$.

Proof. **Step 1:** We recall Corollary C.3.15 in our situation: Suppose $\Delta(X_\lambda, Y_\lambda) \geq \varepsilon(\lambda)$, and let $\delta > 0$, and $\alpha \in [0, \varepsilon]$. Then there is a distinguisher with advantage at least $1 - 2\delta$, which requires

$$N = \lceil 6N_\alpha(\varepsilon - \alpha)^{-2}\log(2\delta^{-1})\rceil$$

samples, where $N_\alpha := \mathsf{tail}^{\dagger}_X(\alpha)$ and has runtime quasi-linear in N (in admissible machine models).

Step 2: Arguing that the statistical distance $\Delta(X, Y)$ is lower-bounded by $1/\mathsf{bnd}$ infinitely often, is not as trivial as in Lemma C.4.30. Indeed, we rely on the general hybrid lemma (Corollary C.3.11) and hence on tail bounds. Suppose the statistical distinguisher has advantage $\geq c$ (infinitely often for constant c). By a standard hybrid argument, Corollary C.3.11, we find a distinguisher which accesses the challenge oracle only once, and has advantage at least

$$\frac{c - \beta}{N_\beta} \quad \text{where} \quad N_\beta := \mathsf{tail}^{\dagger}_{\mathcal{D}_{\mathrm{stat}}}(\beta) \quad \text{for any } \beta \in [0, c].$$

Consequently, $\Delta(X, Y) \geq \frac{c-\beta}{N_\beta}$ for any choice of β. (Note that ε and N_β vary in λ.)

Step 3: Putting Steps 1 and 2 together by (arbitrarily) choosing $\beta = c/2$ we find $\varepsilon = \frac{c}{2N_\beta}$. and $\alpha = \varepsilon/2$ we find

$$N = \lceil 6N_\alpha(\frac{1}{2}\varepsilon)^{-2}\log(2\delta^{-1})\rceil = \lceil 24N_\alpha N_\beta^2 \log(2\delta^{-1})\rceil.$$

[25] To be precise, it is lower-bounded only for infinitely many λ.

Our constructed distinguisher $\mathcal{D}$ needs N samples and has advantage at least $1 - 2\delta$ for infinitely many λ. Now, $N_\alpha = \mathsf{tail}^\dagger_X(\alpha) \in \mathcal{B}$ by assumption that X is $\mathcal{B}$-tailed. Also, $N_\beta = \mathsf{tail}^\dagger_{\mathcal{D}}(\beta) \in \mathcal{B}$ for any constant β, since $\mathcal{D}_{\mathrm{stat}}$ is statistical $\mathcal{T}$-time, hence the number of oracle-queries is $\mathcal{B}$-tailed. Consequently, $N_\alpha N_\beta^2 \in \mathcal{B}$, and we find that $N \in \mathcal{B}$ for any suitable (e.g. constant) advantage $1 - 2\delta$. We obtain a strict $\mathcal{B}$-time distinguisher as promised. ◻

As in Remark C.4.27, one can directly generalize to non-negligible advantage.

Corollary C.4.32. *The result of Corollary C.4.31 extends to the* closure $\overline{\mathcal{T}}$ *of any (suitable) $\mathcal{B}$-time class $\mathcal{T}$. Moreover, it extends to any runtime class in which $\mathcal{T}$ is d-dense.*

C.4.7. From Oracles to Emulation and Standard Indistinguishability

In this section, we abstract properties of runtimes *induced* by algorithms in what we call *continuously samplable.* For such runtimes, we show the equivalence of standard indistinguishability and triple-oracle indistinguishability, which was as introduced for specially runtimes.

Up until now, we treated runtimes as distributions which are samplable via oracle access. This helped keep our options limited and the presentation clean. For applications, we deal with *induced* runtimes of algorithms, and we pay a *non-constant* price for sampling them. To sample the runtime of an algorithm, we *emulate* it. Fortunately, such induced runtimes have a very useful intrinsic property: They are continuously samplable in following sense. To know whether a concrete realization of T is larger than k, we have to emulate at most k steps. If emulation is efficient, and T is efficient, we can therefore sample efficiently. Similarly, if our runtime cutoff bnd is early enough to make $T^{\leq \mathsf{bnd}}$ efficient, then our sampling of $T^{\leq \mathsf{bnd}}$ is efficient. We abstract the central property in the following definition.

Definition C.4.33 (Continuously samplable). A runtime T is **continuously samplable** with overhead function $\mathsf{sampovhd}(k) = \mathsf{sampovhd}_\lambda(k)$, which quantifies the time for sampling T up to time $k \in \mathbb{N}_0 \cup \{\infty\}$; that is: $T^{\leq k}$ can be sampled in $\mathsf{sampovhd}(k)$ steps for all k. More concretely, there is a subroutine $\mathsf{Sample}_T(\lambda, k)$ with output distributed as $T^{\leq k}$ and runtime (strictly) bounded by $\mathsf{sampovhd}(k)$.

We will not specify the overhead $\mathsf{sampovhd}(k)$ and assume it to be "small enough" (e.g. $\mathcal{O}(k\mathsf{polylog}(k))$).[26] In particular, for runtimes induced by algorithms, *sample and emulation overhead essentially coincide if one samples by emulation.* Hence emulation overhead must be small enough to work with the runtime class in question.

Notice that continuous samplability is not tied to any runtime classes per se. In particular, it does not imply efficient samplability without further assumptions.

Example C.4.34. Any runtime which is induced by an algorithm is continuously samplable. Including runtimes of inefficient algorithms.

[26] For PPT, $\mathsf{sampovhd}(\mathsf{poly}_1(\lambda)) \leq \mathsf{poly}_2(\lambda)$ would be good enough. For EPT, emulation requirements are stricter, since runtime may explode under squaring. Interestingly, we reduce only to, and only require, strict algebra-bounded times. Thus, the results in this section do not run into problems with expectation.

Now, we show that for two *continuously samplable* runtimes T, S, where $T \in \mathcal{T}$ (i.e. T is efficient), *oracle-$\mathcal{T}$*-time (in)distinguishable and oracle-*included* $\mathcal{T}$-time (in)distinguishable coincide. This, finally, lets us relate the triple-oracle indistinguishability and standard indistinguishability (under repeated sampling).

Lemma C.4.35. *Suppose that $\mathcal{B}$ is a bound algebra and $\mathcal{T}$ is $\mathcal{B}$-tailed. Suppose that $T \in \mathcal{T}$. Let S be any runtime. Furthermore, suppose that $\mathcal{D}$ is a $\mathcal{T}$-time (triple-oracle) distinguisher with advantage $\geq c$ (infinitely often).*

Then there is a distinguisher $\mathcal{A}$ with advantage $\geq \frac{c}{4}$ (infinitely often) and (a priori) strict oracle-included $\mathcal{B}$-time. More concretely, $\mathsf{time}_{\mathcal{A}}(\mathcal{A}^{\mathcal{O}_0,\mathcal{O}_1,\mathcal{O}^*}) \leq \mathsf{time}_{\mathcal{D}}(\mathcal{D}^{\mathcal{O}_0,\mathcal{O}_1,\mathcal{O}^*})$ up to overhead for emulation and computing the strict bound $\mathsf{bnd}(\lambda)$.

Suppose $\mathcal{A}$ is a distinguisher with runtime strictly bounded by bnd and oracle queries strictly bounded by $\mathsf{bnd}_{\mathsf{query}}$ Suppose T and S are continuously samplable. Then there is an $\mathcal{A}'$ which emulates $\mathcal{O}_0$ and $\mathcal{O}_1$ up to $\mathsf{bnd}_{\mathsf{trunc}} \in \mathcal{B}$ "steps", i.e. emulating $T^{\leq \mathsf{bnd}_{\mathsf{trunc}}}$, $S^{\leq \mathsf{bnd}_{\mathsf{trunc}}}$. By construction, $\mathcal{A}$ is strict $\mathcal{B}$-time with runtime bound roughly $\mathsf{bnd} + 16\mathsf{bnd}_{\mathsf{query}} \cdot \mathsf{bnd}_{\mathsf{trunc}}$ (up to overheads) and advantage at least $\frac{c}{16\mathsf{bnd}_{\mathsf{query}}}$ (infinitely often).

It is vital that $T \in \mathcal{T}$, and hence *efficiently* continuously samplable.

Proof. This first part of the claim is proven analogously to the "standard reduction to PPT", Corollary C.3.9. More concretely: Suppose $\mathcal{O}^* = \mathcal{O}_0$ and consider $\mathcal{D}$. Since $c \in \mathcal{B}$, there exists for some efficiently computable $\mathsf{bnd} \in \mathcal{B}$ because $\mathcal{T}$ is $\mathcal{B}$-tailed. The truncation $\mathcal{A}$ of $\mathcal{D}$ has output with statistical distance at most $\frac{1}{4}c$ (infinitely often). For $\mathcal{O}^* = \mathcal{O}_1$, we either obtain a statistical distance of $\frac{1}{2}c$, or a distinguishing of $\mathcal{O}_0$ and $\mathcal{O}_1$ which uses the runtime statistic $\Pr[S > \mathsf{bnd}] > \frac{1}{2}c$ of $\mathcal{D}$ as distinguishing statistic, just as in Corollary C.3.9. In any case, we obtain $\mathcal{A}$ as claimed.

The second part of the claim follows by definition of continuously samplable and efficiency of T. Namely, let $\mathsf{bnd}_{\mathsf{trunc}}$ so that $\Pr[T > \mathsf{bnd}_{\mathsf{trunc}}] \leq \frac{c}{16\mathsf{bnd}_{\mathsf{query}}}$, where bnd and $\mathsf{bnd}_{\mathsf{query}}$ are strict bounds for runtime and number of queries of $\mathcal{A}$. Since $\mathcal{T}$ is $\mathcal{B}$-tailed and $T \in \mathcal{T}$, an efficiently computable $\mathsf{bnd}_{\mathsf{trunc}} \in \mathcal{B}$ exists. Suppose that $\Pr[S > \mathsf{bnd}_{\mathsf{trunc}}] \leq \frac{c}{8\mathsf{bnd}_{\mathsf{query}}}$. Otherwise, using this distinguishing statistic yields $\mathcal{A}'$ with advantage $\frac{c}{16\mathsf{bnd}_{\mathsf{query}}}$ Now let $\mathcal{A}'$ run $\mathcal{A}$ with the each oracle call to $\mathcal{O}_b$ emulating up to $\mathsf{bnd}_{\mathsf{trunc}}$ "steps" via continuous sampling. The probability that an oracle call returns `timeout` is bounded by $\mathsf{bnd} \cdot \frac{c}{8\mathsf{bnd}_{\mathsf{query}}} = \frac{c}{8}$. In that case, $\mathcal{A}'$ returns a random guess. Thus, $\mathcal{A}'$ has advantage $\frac{c}{8}$ □

Lemma C.4.35 reduces triple-oracle distinguishing to distinguishing w.r.t. repeated samples. It has no requirements on the advantage c of the distinguisher $\mathcal{D}$ and preserves the number of challenge queries in $\mathcal{A}$ and $\mathcal{A}'$. Thus, we can first use a hybrid argument in the triple-oracle setting, reducing to a single challenge query. Then apply Lemma C.4.35. This finally yields the equivalence we wanted.

Corollary C.4.36 (Equivalence of triple-oracle and standard indistinguishability)**.** *Let $\mathcal{B}$ be a bound algebra and $\mathcal{T}$ be $\mathcal{B}$-tailed. Let $T \in \mathcal{T}$ and let S be an arbitrary runtime. Then T and S are* triple-oracle *distinguishable with non-$\mathcal{B}$-negligible advantage, if and only if T and S are* standard *distinguishable with non-$\mathcal{B}$-negligible advantage. (There is $\mathcal{B}$-factor of loss involved in the reduction.)*

Finally, we stress that Corollary C.4.36 is a very loose reduction.

C.5. ⋆ Technical Asides

C.5.1. Section 5.2

C.5.1.1. General Comments

Remark C.5.1 (Unary or binary encodings of runtime). The use of unary encodings in cryptography is more of a compatibility “hack” than a necessity. On the on hand, it is often a convenient “hack”. On the other hand, one has to keep in mind this implicit restriction, and for *statistical* indistinguishability a distinction between binary or unary data is superfluous.

If runtimes were enocded in unary, rather than binary, this would implicity restrict access to a prefix (i.e. a cutoff) which can be efficiently computed. This does not affect our results at all, since our distinguishers and proofs rely on exactly that. Nevertheless, we use binary encoded runtimes and explicit runtime restrictions.

Remark C.5.2 (Non-uniformity and efficiency). Non-uniform (in)security can affect whether an algorithm is considered efficient or not: Suppose there exists an *unkeyed* collision-resistant hash function. An algorithm’s runtime might explode when given colliding inputs. Thus, in the uniform setting, the probability for runtime explosion is negligible, but with non-uniform advice, collisions are trivial. Hence efficiency and security depend on (non-)uniformity. On the other hand, since our results and proofs make only timed black-box use of (adversarial) algorithms, they work in both computational models (with suitable adaptions).

Remark C.5.3 (A priori CPPT). For PPT (and CPPT), the distinction of *a priori PPT* and (a posteriori) PPT is often insignificant. For example, any CPPT algorithm A with virtual runtime bound poly can be truncated to poly steps, giving a (statistically) indistinguishable a priori PPT algorithm A'. Thus, we can usually assume *a priori* PPT for PPT adversaries.

C.5.1.2. Non-Uniform Security

Our proposed notion of non-uniform security is still *probabilistic.* More concretely, we propose in Appendix C.1 to give a probabilistic machine tape-like access to an infinite non-uniform advice string. Usually, non-uniform adversaries are modelled as a priori polynomial time *deterministic* algorithms with advice, or equivalently, polynomial size circuit families. For indistinguishability notions, allowing probabilistic algorithms is typically irrelevant: By standard reductions, a priori PPT adversaries suffice, and so does a priori polynomially bounded advice. By coin-fixing, i.e. fixing the optimal advice and optimal adversarial randomness, one achieves a deterministic a priori non-uniform polynomial time adversary with advantage which is lower-bounded by that of the original (probabilistic) adversary.

Example C.5.4. For oracle-distinguishing, we saw in Corollary 5.4.2, that CEPT distinguishers are no better than a priori PPT distinguishers. For an a priori PPT distinguisher $\mathcal{D}$, it is easy to see that fixing optimal coins yields a deterministic distinguisher $\mathcal{D}'$ with advantage lower-bounded by the advantage of $\mathcal{D}$.

Unfortunately, technical details regarding *preservation of efficiency* still enforce the use of input *distributions.* More concretely, by runtime squaring, there are simulators which are efficient for any input distribution with *strictly* polynomial input size, but become inefficient for distributions with *expected* polynomial input size. see Example C.5.26. Thus, an equivalence with the standard setting of non-uniform security is only guaranteed if security with size-guarding is considered, see Appendix C.5.4.3. While size-guarded security is a natural notion, imposing it when it is not needed is wasteful.

C.5.1.3. Oracle Indistinguishability and Games

We recall a (folklore) conversion between game-based notions and oracle-indistinguishability. Many cryptographic assumption can be cast as (efficient or inefficient) games, in which an adversary interacts with a *challenger* $\mathcal{C}$ (specifying the *experiment* or *game*), and at the end of the interaction, the challenger outputs a verdict $\texttt{win}/\texttt{lose}$ (or 1/0). A hardness assumption is an upper bound for $\Pr[\mathrm{out}_{\mathcal{C}}\langle\mathcal{A},\mathcal{C}\rangle = \texttt{win}]$, e.g. negl for one-wayness or $\frac{1}{2} + \mathsf{negl}$ for IND-CPA, where negl depends on $\mathcal{A}$.

It is generically possible to recast such games as oracle-indistinguishability assumptions: Let $\mathcal{O}_b$ for $b = 0, 1$ be defined as follows. The oracle acts as the game, until the verdict is output. If the verdict is $\texttt{win}$, then $\mathcal{O}_b$ sends b to the adversary. If the verdict is $\texttt{lose}$, then $\mathcal{O}_b$ sends $\perp$ instead. A straightforward calculation shows

$$\Pr[\mathrm{out}_{\mathcal{C}}\langle\mathcal{A},\mathcal{C}\rangle = \texttt{win}] = \Pr[\mathcal{D}^{\mathcal{O}_1} = 1] - \Pr[\mathcal{D}^{\mathcal{O}_0} = 1]$$

where $\mathcal{D}$ is derived from $\mathcal{A}$ by emulating $\mathcal{A}$ until the verdict, and then outputting b (if $\mathcal{A}$ won) or guessing randomly (if $\perp$ was received). Conversely, given $\mathcal{D}$, one defines $\mathcal{A}$ by acting like $\mathcal{D}$ (until the experiment ends). Since information-theoretically, $\mathcal{D}$ obtains learns about b only when it wins the game, $\mathcal{A}$'s success probability is at least that of $\mathcal{D}$. Applying the reverse conversion yields an $\mathcal{D}'$ with advantage equal the probability that $\mathcal{A}$ wins. In other words, both formalizations are equivalent (in any setting that allows the conversion, which encompasses any sensible setting).

The reverse transformation transforms oracle-indistinguishability into a "bit-guessing" experiment. Since the success probability in the experiment is compared to $\frac{1}{2}$, the *advantage* is defined by twice the success probability (so as to coincide with the distinguishing advantage).

C.5.2. Section 5.3

C.5.2.1. More on CEPT

Remark C.5.5 (Non-uniform advice). Observe that the proofs in this section used efficient approximation of (suitably close truncated) distributions as their central tool. This can be effectively trivialized by assuming non-uniform advice, which allows exponentially precise approximations of the truncated distributions, or even simpler, encoding the optimal distinguishing decision for each of the possible samples. Thus, non-uniform advice can replace sampling access, and triple-oracle and standard indistinguishability of runtime coincide (if at least one runtime is efficient).

Remark C.5.6 (Generalizations). The results in this section relied effectively on tail bounds and very natural properties of runtime classes (e.g., if $T \in \mathcal{T}$ and $S \overset{d}{\leq} T$, then $S \in \mathcal{T}$), and . They extend to any setting, where existence of suitable tail bounds is guaranteed. In Appendix C.4, we discuss this more formally, and define *algebra-tailed* runtime classes, to which the results extend in a suitable manner.

We also note that, as already seen in Corollary 5.3.10, it is not necessary that runtime are induced by algorithms, just that they can be approximated up to any polynomial precision in polynomial time. This can be viewed as the central requirement in other proofs and results as well, but it is tedious to formulate and seems only useful in special occasions, e.g., Lemma 5.4.13.

C.5.2.2. More on Timeout Oracles

We continue the discussion of Section 5.3.3 with a specific application to sequential composition.

Lemma C.5.7 (Sequential timeout oracles). *Let* A *be an interactive algorithm and* $\mathcal{O}_1, \mathcal{O}_2$ *be a (probabilistic) timeful oracles. Suppose* $\mathcal{O}$ *is the sequential composition of* $\mathcal{O}_1$ *and* $\mathcal{O}_2$*. That is,* $\mathcal{O}$ *first runs* $\mathcal{O}_1$*. At some point,* $\mathcal{O}_1$ *terminates with input* y *for* $\mathcal{O}_2$*, which is passed to* $\mathcal{O}_2$ *as initial input. Now,* $\mathcal{O}$ *continues to run* $\mathcal{O}_2(y)$*. The results of Lemma 5.3.12 hold for* $\mathcal{O}$*, where* $\Omega_{\mathcal{O}} = \Omega_{\mathcal{O}_1} \times \Omega_{\mathcal{O}_2}$*.*

Moreover the probability ε *for* `timeout` *decomposes as follows: Let event* $\mathcal{E}_{\texttt{timeout},1}$ *be the event for* `timeout` *while running* $\mathcal{O}_1$*. Let event* $\mathcal{E}_{\texttt{timeout},2}(y, t_1)$ *be the event for* `timeout` *while running* $\mathcal{O}_2$ *where* $\mathcal{O}_1$ *took* t_1 *steps to output* y*. Let* Y *and* T_1 *be the random variables for the output and number of steps of* $\mathcal{O}_1$*. Let* $\varepsilon_1 = \Pr[\mathcal{E}_{\texttt{timeout},1}]$ *and let* $\varepsilon_2(y, t_1) = \Pr[\mathcal{E}_{\texttt{timeout},2}(y, t_1) \mid (Y, T_1) = (y, t)]$*. Then*

$$\begin{aligned}
\varepsilon &= \Pr[\mathcal{E}_{\texttt{timeout}}] \\
&= \Pr[\mathcal{E}_{\texttt{timeout},1}] + \sum_{(y,t_1)} \Pr[\mathcal{E}_{\texttt{timeout},2}(y, t_1) \wedge (Y, T_1) = (y, t_1)] \\
&= \varepsilon_1 + \sum_{(y,t_1)} \varepsilon_2(y, t_1) \Pr[(Y, T_1) = (y, t_1)]
\end{aligned}$$

Lemma C.5.7 follows essentially from Lemma 5.3.12 and the fact that the runtimes of $\mathcal{O}_1$ and $\mathcal{O}_2$ sum to that of $\mathcal{O}$. For the decomposition, one argues as in the proof of Lemma 5.3.12, and uses that knowledge of (y, t_1) is good enough for the truncation construction, i.e. $\mathcal{O}_2'$ only needs to know the elapsed time in $\mathcal{O}_1'$ to "continue" the truncation by incorporating the steps of $\mathcal{O}_1'$. The proof is left to the reader.

Remark C.5.8. In the setting of Lemma C.5.7, it is also possible to "separate" $\mathcal{O}_1$ and $\mathcal{O}_2$ instead of treating them as one entity. That is, one can modify them separately, without telling $\mathcal{O}_2$ the runtime t_1 spent in $\mathcal{O}_1$. (Implicit access to t_1 is the only additional knowledge used in Lemma C.5.7.) Concretely, assuming total virtuality ε, one can apply Lemma 5.3.12 for an ε-quantile cutoff to $T_1 = \mathsf{time}_{\mathcal{O}_1}(\mathsf{A}^{\mathcal{O}_1,\mathcal{O}_2})$ to obtain $\mathcal{O}_1'$, and then to $T_2 = \mathsf{time}_{\mathcal{O}_2}(\mathsf{A}^{\mathcal{O}_1',\mathcal{O}_2})$ to obtain $\mathcal{O}_2'$. (For this, note that the virtualities of T_1 and T_2 are certainly bounded by ε.) Together, $(\mathcal{O}_1', \mathcal{O}_2')$ have overall timeout probability of (at most) 2ε and the expected runtime is $t + O(1)$. Unfortunately, the timeout probability of this construction is larger than ε. Except, if y fixes t_1, i.e. if there is a function f such that $f(y) = t_1$, then $\mathcal{O}_1'$ and $\mathcal{O}_2'$ are "separated" by construction (also in Lemma C.5.7).

C.5.3. Section 5.4

C.5.3.1. Precision-Tightness Tradeoff

Most of our results, are very precise in handling the runtime of algorithms, approximating them, and often show that the *distribution* of the runtime only changes negligibly. For example, we proceeded like this in Lemmas 5.3.6, 5.3.8, 5.4.1 and 5.4.7 and Corollaries 5.3.9 and 5.3.10. However, this precision is often unnecessary. Yet, for the sake of simplicity and self-containedness, we always reduced to polynomial support by truncation, and we used a very naive closeness test (see Remark C.5.9 below).

Observe that approximation becomes more expensive (and less tight), the larger the *support* of the distribution is. To improve the state of affairs, one can "coarsen" the time-steps in consideration. Concretely, let $f(x) = 2^{\lceil \log_2(x) \rceil}$, that is, $f(x)$ rounds x to the next power of 2 (and $0 \mapsto 0$). Let X by some positive-valued random variable X (e.g. a runtime). Then we have:

- $\mathbb{E}[X] \leq \mathbb{E}[f(X)] \leq 2 \cdot \mathbb{E}[X]$, since $x \leq f(x) < 2x$.
- $\mathrm{card}(\mathrm{supp}(f(X))) \leq \log_2(\mathrm{card}(\mathrm{supp}(X)))$

In particular, consider an EPT or CEPT runtime T, and assume that $T_\lambda \leq 2^\lambda$. Then $\mathrm{card}(\mathrm{supp}(f(T))) \leq \lambda$, whereas $\mathrm{card}(\mathrm{supp}(T^{\leq \mathsf{poly}})) = \mathrm{poly}(\lambda)$, for any polynomial poly. Thus, the coarser $f(T)$ is, the more efficient approximation (and closeness testing) becomes. However, precision is lost in the time-domain. Generally, this is irrelevant, since efficiency is not affected at all by this uncertainty.

Remark C.5.9 (Closeness testing). Our analysis was not geared towards optimal tightness and precision to begin with. And, for simplicity, we actually never made explicit, that we often merely require *closeness tests* for distributions (see Appendix C.3.4 for a reminder). We used a naive approximation of distributions as our closeness test, but there are much better alternatives. However, even for the (optimal) closeness tester of [CDVV14], its precision depends on the size of the support. Hence, using [CDVV14] further improves in tightness, is overall is still very loose.

C.5.3.2. Constructive Reductions

While we prove all of our results in a uniform complexity setting, hence avoid non-uniformity, almost none of our proofs are constructive. In fact, almost all reductions depend on polyomial bounds, which exist but are not computable in general. We will use following rough notion of constructive reductions.

Remark C.5.10 (Constructive reduction). A reduction is *constructive*, if there is a (universal) efficient algorithm, which is given the code of an adversary, and produces (the code of) an adversary against some underlying assumption.

For simple results, such as the standard truncation argument, one can give constructive proofs in restricted cases. We sketch how these can be obtained.

Remark C.5.11. A weakening of the standard reduction (Lemma 5.4.1 and Corollary 5.4.2) can be proven constructively. We sketch how to transform the distinguisher $\mathcal{D}$ into a distinguisher $\mathcal{A}$ (where $\mathcal{D}$, $\mathcal{O}_0$, $\mathcal{O}_1$ are as the standard reduction). For the constructive $\mathcal{D}$, we have to merge the two distinguishers $\mathcal{A}_1$ (which uses $\mathtt{timeout}$) and $\mathcal{A}_2$ (which uses the output of $\mathcal{D}$). One problem is, that the use of "unsigned" advantage (i.e. absolute values instead of the (signed) differences) does not work well constructively, since we cannot "know" the signs. Thus, we use standard sign-correction techniques. Consider some distinguisher $\mathcal{B}$. To sign-correct, the reduction runs $\mathcal{B}^{\mathcal{O}_0}$ and $\mathcal{B}^{\mathcal{O}_1}$, emulating $\mathcal{O}_0$ and $\mathcal{O}_1$, to get output (and runtime statistics). Using this, one corrects the output of $\mathcal{B}^{\mathcal{O}^*}$, so that $\Pr[\mathcal{B}^{\mathcal{O}_b} = b] \geq \frac{1}{2}$ for $b = 0, 1$. This gives us a first restriction: We require that emulating $\mathcal{B}^{\mathcal{O}_b}$ is efficient for $\mathcal{O}_0$ and $\mathcal{O}_1$. In particular, $\mathcal{O}_0$ and $\mathcal{O}_1$ will have to be efficient in some sense.

Consider some efficiently samplable runtime distribution S (to be chosen later). Our distinguisher $\mathcal{A}$ works as follows:

- Pick $s \leftarrow S$.
- Set $\mathcal{B} := \mathcal{D}^{\leq s}$.
- Emulate $\mathcal{B}^{\mathcal{O}_b}$ to obtain runtimes t_b and outputs out_b for $b = 0, 1$. (We assume $out \in \{0, 1\}$.)
- Emulate $\mathcal{B}^{\mathcal{O}^*}$ and obtain (t^*, out^*) and output a guess as follows:
  - If $t_b = \mathtt{timeout} \neq t_{1-b}$: If $t^* = \mathtt{timeout}$ return b, else return $1 - b$.

- If $t_0, t_1 \neq \texttt{timeout}$ and $out_0 \neq out_1$: Return out^* xor $(out_0$ xor out_1 xor $1)$.
- Else: Return 0.

By construction, $\mathcal{A}$ is a merge of a distinguisher based on $\texttt{timeout}$ probabilites and a distinguisher based on the output of $\mathcal{D}$. Both are sign-corrected, so the advantages actually add up (and don't cancel out). To guarantee non-negligible advantage, when $\mathcal{D}$ has non-negligible advantage, we must ensure that the truncation $\mathcal{B}$ of $\mathcal{D}$ at s gives $\mathcal{B}$ enough runtime with polynomial probability. For this, one can use any distribution S which is EPT and has fat tails, e.g. the distribution obtained from normalizing $\sum_{n=1}^{\infty} n^{-3}$. However, for $\mathcal{A}$ to be efficient overall, we additionally require the time spent to emulate $\mathcal{O}_0$ and $\mathcal{O}_1$ does not make $\mathcal{A}$ inefficient (for the varying $s \leftarrow S$). One possibility is to restrict to a priori PPT $\mathcal{O}_0$, $\mathcal{O}_1$, but less strict choices are possible.[27] All in all, this yields a weaker, less tight, more restricted, but constructive, form of the standard reduction.

C.5.3.3. Relative Efficiency for Mappings of Systems

The definition of weak relative efficiently (Definition 5.4.5) does not strictly capture our actual application. Indeed, a simulator takes as *input* an adversary, i.e. a system/oracle (or an algorithm/code), and *acts* as (or *outputs*) a new system. Hence, (the existence of) a simulator is actually a mapping from admissible adversaries to simulators. This is quite obvious for universal (resp. bb-rw) simulation, where the code (resp. bb-rw access) are clear "inputs". Since the simulator is independent of the input generation or the distinguisher, i.e. of the "distinguishing environment", it is also evident that $\mathsf{Sim}(\mathsf{code}(\mathsf{V}^*))$ has an input and output interface. The input is $(\mathbb{x}, \mathbb{w}, aux)$, the output is the output of V^*. While Sim discards $\mathbb{w}$, it is necessary so that $\mathsf{Sim}(\mathsf{V}^*)$ and $\langle \mathsf{P}, \mathsf{V}^* \rangle$ offer the same interface to the environment. We sketch a general definition of the above.

Definition C.5.12. A **mapping** of systems (or algorithms) to systems (or algorithms) is a function $\mathrm{F}\colon C_I \to D_J$ with maps system (or algorithms) with interface I to systems (or algorithms) with interface J.

With this, we can define when a mapping G is weakly efficient relative to a mapping F. This generalizes Definition 5.4.5, which can be recovered from the constant mappings $\mathrm{F} = \mathsf{A}$, $\mathrm{G} = \mathsf{B}$.

Definition C.5.13. Let $\mathrm{F}, \mathrm{G}\colon C_I \to D_J$ be mappings. We say G is **weakly $(\mathcal{T}, \mathcal{S})$-efficient relative to** F *w.r.t. (implicit) runtime classes $\mathcal{T}$, $\mathcal{S}$*, if for all distinguishing environments $\mathcal{E}$,

$$\forall \mathcal{A} \in C_I\colon \quad \mathsf{time}_{\mathcal{E}+\mathrm{F}(\mathcal{A})}(\langle \mathcal{E}, \mathrm{F}(\mathcal{A}) \rangle) \in \mathcal{T} \implies \mathsf{time}_{\mathcal{E}+\mathrm{G}(\mathcal{A})}(\langle \mathcal{E}, \mathrm{G}(\mathcal{A}) \rangle) \in \mathcal{S}$$

Unfortunately, Definition C.5.13 is not strong enough to be used in reductions, which is why we reserved the specification "*weakly*" for Definitions 5.4.5 and C.5.13 (More precisely, we cannot prove or refute that it is (not) strong enough.) Therefore, we rely on following strengthening, where, the runtime classes $\mathcal{T}$ and $\mathcal{S}$ are from $\{\mathcal{PPT}, \mathcal{EPT}, \mathcal{CPPT}, \mathcal{CEPT}\}$, and they decide whether strict or expected time is measured.[28]

[27] As seen with the runtime squaring problem and expected polynomial size inputs, relatively stringent requirements on $\mathcal{O}_0$, $\mathcal{O}_1$ (or size-guarding, c.f. Appendix C.5.4.5) seem necessary in general.
[28] This generalizes to other norms besides $\|\cdot\|_1$ and $\|\cdot\|_\infty$ as measures of efficiency.

Definition C.5.14. Let F, G, etc. be as in Definition C.5.13. We say that G is **$(\mathcal{T}, \mathcal{S})$-efficient relative** to F with **runtime tightness** $(\mathsf{poly}_{\mathsf{time}}, \mathsf{poly}_{\mathsf{virt}})$, if: For *all timeful* environments $\mathcal{E}$ and all $\mathcal{A} \in C_I$, if $\mathsf{time}_{\mathrm{F}(\mathcal{A})}(\langle \mathcal{E}, \mathrm{F}(\mathcal{A})\rangle)$ is virtually strict/expected (t_0, ε_0)-time, then $\mathsf{time}_{\mathrm{F}(\mathcal{A})}(\langle \mathcal{E}, \mathrm{G}(\mathcal{A})\rangle)$ is virtually strict/expected (t_1, ε_1)-time, with $t_1(\lambda) \le \mathsf{poly}_{\mathsf{time}}(\lambda) t_0(\lambda)$ with $\varepsilon_1(\lambda) \le \mathsf{poly}_{\mathsf{virt}}(\lambda)\varepsilon_0(\lambda)$ (for all λ).

We stress that Definition C.5.14 is unconditional w.r.t. the environment, i.e. uses timeful environments, and that the tightness bounds depend *only on* λ. Mixing strict and expected time (i.e. $\|\cdot\|_\infty$ and $\|\cdot\|_1$) in Definition C.5.14 is possible and useful. For example when strict PPT protocols and adversaries are handled by EPT simulators.

(Weak) Relative efficiency is transitive in the obvious sense. Lastly, we mention that there are obvious variations of relative efficiency, e.g. relative efficiency w.r.t. environments in a class $\mathfrak{E}$ of admissible environments with restriction beyond runtime.

C.5.4. Section 5.5

In this section, we discuss some technical asides. It should be skipped on a first reading.

C.5.4.1. Definitional Choices

Remark C.5.15 (The adversary's view). We did not use the *view* of the adversary to define zero-knowledge for a reason. The usual definition of a view consists of input, randomness, and received messages. This conflates different complexities, e.g. randomness and space, and thus prevents *strict polynomial space simulation*, which may be of interest. For example, the simulator for $\mathrm{G3C}_{\mathrm{GK}}$ uses expected polynomial randomness and space, even if V^* requires only strict polynomial space (since $\mathsf{bbrw}(\mathsf{V}^*)$ chooses and fixes (i.e. remembers) the random coins) A slightly "improved" simulation requires only strict polynomial space. For more discussion on interaction of (bb-rw) emulation with strict versus expected space and randomness complexity, see Remark C.1.6.

Remark C.5.16. By a standard reduction to PPT (Corollary 5.4.2), we can assume that $\mathcal{D}$ is a priori PPT in Definition 5.5.2. A formulation of zero-knowledge via indistinguishable ensembles,

$$\begin{aligned} &\{(\mathbb{x}, \mathit{aux}, \mathit{out}, \mathit{state}) \mid (\mathbb{x}, \mathbb{w}, \mathit{aux}, \mathit{state}) \leftarrow \mathcal{I}(\lambda); \mathit{out} \leftarrow \mathsf{out}_{\mathsf{V}^*}\langle \mathsf{P}(\mathbb{w}), \mathsf{V}^*(\mathit{aux})\rangle(\mathbb{x})\}_\lambda \\ \overset{c}{\approx}\ &\{(\mathbb{x}, \mathit{aux}, \mathit{out}, \mathit{state}) \mid (\mathbb{x}, \mathbb{w}, \mathit{aux}, \mathit{state}) \leftarrow \mathcal{I}(\lambda); \mathit{out} \leftarrow \mathsf{Sim}(\mathsf{code}(\mathsf{V}^*), \mathbb{x}, \mathit{aux})\}_\lambda \end{aligned}$$

is then equivalent to Definition 5.5.2.

Remark C.5.17. There are other formulations of zero-knowledge, which can be obtained by swapping the order of the quantifiers. To recover the "usual notions" (that is universal quantification over the inputs), $\mathcal{I}$ should be instantiated by a non-uniform machine which regurgitates its advice.

(Timed) Black-box simulator: Timed bb-rw access to V^*. Most common form of simulation.

Universal simulator: $\exists \mathsf{Sim} \forall \mathsf{V}^* \forall \mathcal{I} \forall \mathcal{D}$. Typical form of non-black-box simulation, e.g. in [Bar01].

Existential simulator: $\forall \mathsf{V}^* \exists \mathsf{Sim} \forall \mathcal{I} \forall \mathcal{D}$. Typical definition of zero-knowledge, e.g. in [Gol01].

We see in Appendix C.5.4.4 below that existential simulation and universal simulation are equivalent for *auxiliary input* zero-knowledge for *a posteriori time.*

There are also less common, weaker notions, such as *distributional simulation* (roughly "$\forall \mathsf{V}^* \forall \mathcal{I} \exists \mathsf{Sim} \forall \mathcal{D}$"), *weak simulation* (roughly "$\forall \mathsf{V}^* \forall \mathcal{D} \exists \mathsf{Sim} \forall \mathcal{I}$"), *weak distributional simulation* (roughly "$\forall \mathsf{V}^* \forall \mathcal{D} \forall \mathcal{I} \exists \mathsf{Sim}$"), see [DNRS03; CLP15]. Likewise, there are stronger notions, such as precise zero-knowledge [MP06; DG12] where simulation and real execution must have similar runtime per execution. We have not pursued an adaption to CEPT for these notions.

Remark C.5.18 (Non-uniform zero-knowledge). When considering non-uniformity, there are different options. In any case, $\mathcal{I}$ and $\mathcal{D}$ (equivalently $\mathcal{E}$) should be non-uniform. Now, V^* can be uniform or non-uniform. For Sim, there are two similar options. One is to insists on uniform simulation, in the sense that the advice Sim is given the same as the advice of V^*. The other is to allow an existential non-uniform Sim, whose advice may arbitrarily depend on V^* and $\mathit{advc}_{\mathsf{V}^*}$. Goldreich [Gol01] calls the latter a "fully non-uniform" formulation of zero-knowledge, and argues that the former is preferable. These choices do not affect our results based on black-box simulation. However, the existence of a universal $\mathcal{V}_{\text{univ}}$ is unclear, if V^* has access to non-uniform advice.

Remark C.5.19 (Effects of "(non-)environmental" distinguishing). Let us consider the effect of "non-environmental" environments $(\mathcal{I}, \mathcal{A}, \mathcal{D})$, i.e. quantification over $\mathcal{I}$ which output no *state* (i.e *state* $= \perp$ always). In this case case, Sim has the same information that is available to $\mathcal{D}$, whereas in Definition 5.5.2 $\mathcal{I}$ can pass information to $\mathcal{D}$ directly.

Since a bb-rw simulator has no access to *aux*, *aux* can be used to pass "direct" messages to $\mathcal{D}$, thus both notions coincide in this case. It seems plausible, that simulators which do not "reverse-engineer" V^* and *aux* and are "non-environmentally" secure are also "environmentally" secure, i.e. we know of no counterexamples even for non-black-box simulators. Intuitively, $\mathcal{E}$ and $\mathcal{D}$ may share a "key" (e.g. as non-uniform advice or hardwired), and use a one-time pad to "encrypt" messages which are passed from $\mathcal{E}$ to $\mathcal{D}$. It is easy to see that, if Sim is not secure, then there is a (sequence of) "keys", such that the advantage of a suitable non-environmental $(\mathcal{E}', \mathcal{D})$ is at least that of $\mathcal{E}$ (infinitely often), if the "key" is long enough to one-time pad "encrypt" the state of $\mathcal{E}$ passed between input generation and distinguishing. (Summing the advantage over hardwired or non-uniform key k for $(\mathcal{E}'_k, \mathcal{D}_k)$ over all possible $\mathsf{poly}(\lambda)$-bit keys, weighted by $2^{-\mathsf{poly}(\lambda)}$ is exactly the advantage of $\mathcal{E}$. The claim follows.) Thus, in a uniform model, constant size messages can be passed to $\mathcal{D}$ without affecting security, and in a non-uniform model, polynomial size messages can be passed.

Remark C.5.20 ("Environmental" distinguishing and non-uniformity). The additional output *state* of $\mathcal{I}$ in Definition 5.5.2 essentially makes $(\mathcal{I}, \mathcal{D})$ into a stateful distinguishing "environment". This is a visible difference from the direct translation of non-uniform zero-knowledge and [Gol93]

In a non-uniform CEPT setting, Definition 5.5.2 does coincide with the standard definition, if $\mathcal{I}$ and $\mathcal{D}$ have non-uniform advice. To see this, observe that we can assume that successful $\mathcal{I}$ and $\mathcal{D}$ are a priori PPT, and by coin-fixing (i.e. fixing the optimal coins for $\mathcal{I}$ and $\mathcal{D}$ in the advice), they are deterministic. Now, *state* can simply be included in the non-uniform advice of $\mathcal{D}$ as well. Hence, in this non-uniform PPT setting, the notions are equivalent.

More generally, if a successful $\mathcal{I}$ only chooses $(\mathtt{x}, \mathit{aux})$ of strictly polynomially bounded size, then there is an optimal choice and adaptions of the coin-fixing argument work. By Example C.5.26, we know that for covering simulation efficiency, we cannot restrict to such strictly polynomially bounded $(\mathtt{x}, \mathtt{w}, \mathit{aux}, \mathit{state})$. Perhaps, this can be salvaged this somewhat. However, we find such "non-uniformity hacks" very unsatisfactory (even if they work). They result in "less robust" definitions (which is the reason we had to adapt the definitions in the first place).

Remark C.5.21 (Simulation tightness and inefficient provers). In Definition 5.5.2, we compare the runtime of a simulator with the runtime of V^* *and* P. We do so, because efficiency (and tightness) of a simulation should be related to efficiency of the real execution.[29] Alternatively, we could compare $\mathsf{time}_{\mathsf{V}^*+\mathsf{P}}(\ldots)$ and $\mathsf{time}_{\mathsf{Sim}}(\ldots)$. This is equivalent, if the completed system with $\mathcal{I}, \mathcal{D}$ was efficient.[30]

By viewing P as timeful and setting its runtime to the length of messages sent by it (which is the minimal consistent choice for time), Definition 5.5.2 extends to inefficient provers. However, technical artefacts occur, e.g. if a simulation must run in quadratic time, then inputs which are *expected* polynomial size, can cause runtime explosions, and therefore the proof system is not zero-knowledge. If such problems occur, they can usually be circumvented by resorted to size-guarded security, which ensures that inputs are strictly polynomially bounded.[31]

Remark C.5.22 (Precomputation and different complexity classes). Non-uniform advice is often motivated as a means to strengthen attacks and allow arbitrary (even uncomputable) "precomputation". However, the practical meaning of non-uniformity is questionable [KM13]. Fortunately, precomputation neatly fits into our model by using different complexity classes for input generators (and possibly distinguishers or environments). This can be applied to our definitions of (sequential) zero-knowledge, but was omitted for the sake simplicity.

C.5.4.2. Motivating Sequential Zero-Knowledge

Often, e.g. in [Gol93], only sequential repetition is considered for zero-knowledge. That is, the same pair $(x, w) \in \mathcal{R}$ is used in multiple rounds of the interactive argument. The (stateful) adversary V^* engages in these multiple rounds until it produces some output. To obtain zero-knowledge for such a sequential repetition, the core property is the *auxiliary input.* Construct from V^* a new V' which simply executes the code of V^* on *aux*. (More precisely, *aux* encodes either a state of V^* or, in the first invocation of V', it is the actual auxiliary input for V^*.) Thus, each protocol now runs with the same adversary V', but different auxiliary inputs. Applying auxiliary input zero-knowledge and a hybrid argument, one sees that each interaction can be simulated.

In our setting, the universal adversary $\mathcal{V}_{\text{univ}}$ could be used directly (due to a posteriori time). Specifying and proving security of sequential repetition for *varying* statements is also possible along these lines. However, for *adaptive choices* of varying statements, it's not clear how to allow it with a "single" adversary V^*. If V^* provides (x, w) then simulation becomes meaningless. Thus, we introduce an "environment", which models the use of the protocol. The environment $\mathcal{E}$ can make repeated calls to the protocol and choose the inputs (for both parties) and an adversary V^*. Then $\mathcal{E}$ obtains the output of V^* (since P has no output). Finally, $\mathcal{E}$ produces some output. The "environment" does not participate in the computation. It is essentially a distinguisher for sequential real or ideal protocol executions.

Remark C.5.23. The weaker notion of sequential repetition, as sketched above, follows from auxiliary input zero-knowledge, even if the "environmental" $\mathcal{I}$ is not allowed not output *state*, e.g. if always

[29] This entails some technical artefacts, e.g. a prover may be badly behaved for invalid inputs, e.g. not halting. The complexity class for "good protocols" should be robust and prevent such behaviour.

[30] However, $\mathsf{time}_{\mathsf{V}^*+\mathsf{P}}(\ldots)$ and $\mathsf{time}_{\mathsf{Sim}}(\ldots)$ are more suitable for analyzing the tightness of simulation, which we did not define (since we do not have a perfectly convincing definition).

[31] An alternative "fix" is to prevent too efficient verifiers, e.g. using timelock puzzles.

state = $\bot$. This is in line with [Gol93], and uses that everything the adversary "knows" the simulator "knows". However, for two (or more) adaptive statements, the "environment" has "knowledge" which the simulator has not. In the classical definition (with classical PPT), the power of non-uniformity still allows to prove sequential composition. The non-uniform advice of $\mathcal{I}$ and $\mathcal{D}$ effectively establishes the "shared *state*" between $\mathcal{I}$ and $\mathcal{D}$.

C.5.4.3. Size-Guards and Size-Guarded Security

In our definition of zero-knowledge, due to fat tailed input distributions, a simulator is allowed almost no runtime overhead in $|\mathbb{x}|$ compared with P, i.e. if a prover is linear-time in $|\mathbb{x}|$ the simulator must also be. In [KL08; Gol10], the simplification $\lambda = |\mathbb{x}|$ is used, which alleviates this issue somewhat. We mirror that by explicitly *size-guarding* a protocol. This means that prover (and verifier) reject inputs which exceed the length of a (polynomial) size-guard $\mathrm{gd}(\lambda)$. Size-guards "decouple" efficiency of simulator and prover w.r.t. $|\mathbb{x}|$, simplify efficiency arguments, but slightly weaken security.

Definition C.5.24 (Size-guarded zero-knowledge). We define **(uniform) zero-knowledge w.r.t. (input) size-guarded** security as follows: For any (monotone) polynomial bound gd (called **size-guard**), the derived protocol, where prover and verifier abort with $\mathtt{gderr}$ on inputs $(\mathbb{x}, \mathbb{w})$ if $|\mathbb{x}| > \mathrm{gd}(\lambda)$, is zero-knowledge (in the above sense).

The definition of **non-uniform** (size-guarded) zero-knowledge is analogous to the above, but $\mathcal{I}(\lambda)$ has access to an advice via an additional input interface.

Definition C.5.25. We define **(non-)uniform sequential zero-knowledge w.r.t. (input) size-guarded** security (see Definition C.5.24) as follows: For any polynomial size-guard gd, the derived protocol, where prover and verifier abort with $\mathtt{gderr}$ on inputs $(\mathbb{x}, \mathbb{w})$ where $|\mathbb{x}| > \mathrm{gd}(\lambda)$, is sequential zero-knowledge (in the above sense).

The use of (input) size-guarded security is meant to address certain situations, which we may want to consider secure, but cannot due to runtime artefacts.

Example C.5.26. Namely, suppose the simulator has quadratic runtime in the instance size $|\mathbb{x}|$, whereas the prover's runtime is linear. Then, a problematic fat-tailed input distribution renders simulation inefficient. Consequently, without size-guarding, simulation must be "tight" in $|\mathbb{x}|$. One technical artefact, partially mitigated by size-guards, is that very efficient provers make simulation harder. That is, by making a prover slower, e.g. adding a quadratic overhead, simulation becomes easier.

These problems may be of practical relevance: Given succinct argument systems, extraction comes with an overhead which is often superlinear. Such argument systems can be incompatible with CEPT under adversarial input distributions.

There are other means than size-guarding for solving the above problem. For example, one may quantify only over admissible adversaries. Indeed, adversaries which only send strictly polynomial size inputs are equivalent to size-guarded security.

See Appendix C.5.4.5 for more on size-guards.

C.5.4.4. The Universal Adversary $\mathcal{V}_{\text{univ}}$

The **universal adversary** $\mathcal{V}_{\text{univ}}$ is basically a virtual machine emulating some adversary, i.e. the input to $\mathcal{V}_{\text{univ}}$ is of the form $(\mathit{code}, \mathit{state}, \mathit{aux})$, and $\mathcal{V}_{\text{univ}}$ continues execution of the code *code* in state *state*. The universal adversary $\mathcal{V}_{\text{univ}}$ contains the core hardness of simulation. An **existential** simulator is a simulator which may depend arbitrarily V^*. The universal adversary shows that this arbitrary "existential" dependency on V^* does not weaken the notion of zero-knowledge. Thus, in Definition 5.5.2, we do not give up any power.

Lemma C.5.27 (Equivalence of existential and universal simulation)**.** *Let* (P, V) *be an interactive argument system. If this argument system is zero-knowledge against* $\mathcal{T}$*-time designated adversaries w.r.t. to* $\mathcal{S}$*-time existential simulation, then it is zero-knowledge w.r.t. the universal simulator* Sim *defined as follows.*

Let $\mathcal{V}_{\text{univ}}$ *be the universal adversary and* $\mathsf{Sim}_{\text{univ}}$ *be the existential simulator for* $\mathcal{V}_{\text{univ}}$*. Here,* Sim *is defined by* $\mathsf{Sim}(\mathrm{code}(\mathsf{V}^*), \mathbb{x}, \mathit{aux})$ *emulating* $\mathsf{Sim}_{\text{univ}}(\mathrm{code}(\mathcal{V}_{\text{univ}}), \mathbb{x}, (\mathrm{code}(\mathsf{V}^*), \mathit{state}, \mathit{aux}))$*, where state is the initial state of* V^**.*

Proof. First we define $\mathcal{G}_{\mathsf{V}^*}$, which samples $(\mathbb{x}, \mathbb{w}, \mathit{aux}) \leftarrow \mathcal{G}$, and returns $(\mathbb{x}, \mathbb{w}, (\mathrm{code}(\mathsf{V}^*)), \mathit{state}, \mathit{aux})$, where *state* is the initial state of V^*. Recall that for $(\mathcal{G}, \mathsf{V}^*)$, the simulator $\mathsf{Sim}(\mathrm{code}(\mathsf{V}^*), \mathbb{x}, \mathit{aux})$ runs $\mathsf{Sim}_{\text{univ}}(\mathrm{code}(\mathcal{V}_{\text{univ}}), \mathbb{x}, (\mathrm{code}(\mathsf{V}^*), \mathit{state}, \mathit{aux}))$, which corresponds to $(\mathcal{G}_{\mathsf{V}^*}, \mathcal{V}_{\text{univ}})$. Moreover the real executions $\mathsf{Real}_{\mathcal{G},\mathsf{V}^*}$, and $\mathsf{Real}_{\mathcal{G}_{\mathsf{V}^*},\mathcal{V}_{\text{univ}}}$ are identical. Thus

$$\mathsf{Ideal}_{\mathcal{G},\mathsf{Sim}(\mathsf{V}^*)} \overset{d}{\equiv} \mathsf{Ideal}_{\mathcal{G}_{\mathsf{V}^*},\mathsf{Sim}_{\text{univ}}} \overset{c}{\approx} \mathsf{Real}_{\mathcal{G}_{\mathsf{V}^*},\mathcal{V}_{\text{univ}}} \overset{d}{\equiv} \mathsf{Real}_{\mathcal{G},\mathsf{V}^*}.$$

□

The upshot of the proof is, that an existential simulator cannot truly leverage its arbitrary dependency on V^*. All the hardness of V^* might be in the *auxiliary input*, which Sim cannot depend upon. Lemma C.5.27 extends to the non-uniform setting and to size-guarding.

Caution C.5.28. We crucially relied on our *a posteriori* runtime notion. For other notions of runtime, Lemma C.5.27 may not hold! For example, if we assume *a priori* PPT algorithms, then $\mathcal{V}_{\text{univ}}$ cannot emulate every adversary V^*, since $\mathcal{V}_{\text{univ}}$ must not exceed poly steps, for some *fixed* poly, whereas V^* may run (much) longer, say poly + 1 steps. (There is a family $\mathcal{V}^n_{\text{univ}}$ with runtime bounds $\mathrm{poly}_n(\lambda) = n\lambda^n$, so a morally equivalent result does hold.)

C.5.4.5. Size-Guards

We recall the need for size-guards, discuss two approaches to generalizing size-guarding, and mention complexity classes for which size-guarding is superfluous, Then, we identify some problems with size-guards, which may complicate their use. For we generality, consider a generic real-ideal setting, and use zero-knowledge as an example. It is easy to see that both proposed notions of size-guarding are efficient transformations (in any sensible machine model).[32]

[32] The effect of size-guarding on runtime is minor. If a lazy size-guard implementation is used, instead of eagerly checking the size, then up to emulation overhead, the runtime doubles at most. (Eager implementations may blow up runtime if the time for writing the message is not accounted for, e.g. because of huge messages from (inefficient) oracles.)

A Case for Size-Guards. Recall that adversarial input *distributions*, which exploit expected polynomial size via fat-tailed distributions, may yield simulators which are not CEPT, because the have a, say quadratic, dependency on input length, whereas the real protocol (e.g. the prover) has a linear dependency, see Example C.5.26. Bounding input length, or even message length, which honest parties accept hardly affects the usefulness of a protocol. Indeed, these bounds are fixed *a posteriori*, i.e. after the full system is built from its parts. We have no good example for a setting, where there is no suitable polynomial bound one the input (or message) length of honest parties. So we expect that such a posteriori restrictions do not affect real applications.

Size-Guarding Inputs. The most natural approach to size-guarding is arguably to size-guard inputs to ideal functionalities, i.e. messages sent to the interface of real protocol or their ideal equivalent. Size-guarding a functionality yields a new functionality, which aborts upon receiving inputs which exceed the length allowed by the size-guard. (Adversarial parties should be allowed to ignore size-guard restrictions. Also, other parties should be notified of such an abort.) As explained above, we know no good example where a functionality cannot be replaced in such a way.

This simplistic sketch of size-guarding may be ill-defined, and lead to problems, as in pointed out in a later paragraph.

Size-Guarding Communication. Instead of size-guarding only inputs, one may want to size-guard all communication of honest parties. This may also be viewed as size-guarding all interfaces and the communication channel. (We should not impose size-guards on adversarial communication, as there is no justification for limiting their communication.) This kind of size-guarding is formally stronger, but we expect that for most (all?) interesting protocols, it is equivalent with size-guarding inputs. However, size-guarding communication affects everything, not just functionalities. Thus, we find the more local notion of size-guarding inputs preferable, and less likely to lead to unpleasant surprises.

Strict Polynomial Space. An algorithm A has a priori *strict (probabilistic) polynomial space* (SPS) (in analogy to PPT) if there exists a polynomial $\mathsf{poly}(\lambda)$ which bounds maximal used space/memory. We count outgoing (but not incoming) message queues as part of an algorithms space/memory. With this, any size-guard larger than poly does not affect the behaviour of A at all. Thus, for a priori SPS adversaries, size-guarded security and normal security are equivalent.

For "classical" SPS, the space of A may depend on the input size, i.e. $\mathsf{poly}(\lambda, |x|)$. All protocols of interest satisfy SPS. We find (classical, a posteriori and a priori) EPT SPS algorithms an appealing complexity class. Of course, the "negligible slack" of CEPT and CPPT is motivated for and applies to this setting as well. Unfortunately, deterministic bb-rw oracles for EPT adversaries are not compatible with SPS, hence our simulators are not PPT either. This can likely be fixed, see Remark C.1.6.

Composability and Definitional Issues. One major drawback of size-guarded security, is that it *changes the ideal functionality*. This may break properties, such as correctness, of protocols using such subprotocols. As mentioned before, size-guards should be chosen after a system is composed, so that such problems do not occur. Since a protocol may call a subprotocol with squared input length, for composition, one needs to keep track of size-guards, and be aware that they may not be identical for all protocols. That is, a protocol which is built from subprotocols imposes different size-guards on the subprotocols than the size-guard which was imposed on itself.

Another problem of size-guards is, that it may be convenient or relevant to have different or more fine-grained size-guards for different interfaces. E.g. for zero-knowledge, we left the witness unguarded. A more flexible approach than merely limiting the input length may be useful in a larger setting.

An Alternative to Size-Guards. The problems noted above seem to disappear if instead of size-guarding inputs and changing protocol behaviour, one restricts to "admissible adversaries", as mentioned in Example C.5.26. The drawback is that now, one needs to specify admissibility variations for all notions, e.g. rewinding strategies, relative efficiency, and so on. We also caution that, similar problems as for size-guards may appear, just hidden deeper in security proofs.

C.5.5. Section 5.6

A simulator Sim is **benign under size-guarding**, if it is benign (Definition 5.6.20) whenever a polynomial size-guard is imposed on the protocol. Analogous claims for Lemmas 5.6.23 and 5.6.26 hold w.r.t. size-guarded zero-knowledge.

C.5.5.1. Connection Between Runtime and Probability Tightness

Following example illustrates, that normality is not automatic.

Example C.5.29 (Bad RWS). Consider a proof system with a (useless) preamble, where the prover sends a random string $s \xleftarrow{\$} \{0,1\}^\lambda$, the verifier acknowledges it, and the actual protocol begins. A rewinding strategy RWS could send 0^λ as its first query, and then rewind. Against classical PPT adversaries, this is no problem at all. However, this essentially notifies the adversary of being in a simulation. Indeed, the probability tightness of RWS is 2^λ. Similarly, a rewinding strategy RWS, which "prefers" to output lexicographically smaller transcripts, typically has (very) noticeable output skew.

For EPT adversaries, runtime tightness implies probability tightness asymptotically.

Remark C.5.30 (Necessity of probability tightness). Let RWS be a rewinding strategy for (P, V). Let V^* be a deterministic malicious verifier. Suppose there is a (sequence of) logical queries $query = query(\lambda)$ such that $\mathrm{pr}_{\mathrm{rws}}(query) > \frac{1}{\mathrm{negl}} \cdot \mathrm{pr}_{\mathrm{real}}(query)$ infinitely often. By modifying V^* to run an extra $\frac{1}{\mathrm{pr}_{\mathrm{real}}(query)}$ steps if queried with $query$, we obtain a new deterministic verifier V^{**} whose expected runtime increases by 1. But RWS incurs a superpolynomial runtime growth, as it cannot see the "trap". Thus, runtime tightness implies probability tightness.

The "attack idea" on RWS in Remark C.5.30 may also be viewed as an indication that almost(?) all rewinding strategies in the literature are normal. More generally, even an a priori PPT adversary can exploit a large probability tightness and give "bad" answers in such cases. So, even for a priori PPT adversaries which cannot cause runtime explosion, there is no incentive to have large probability tightness, because it is unclear how that could be usefully exploited.

Remark C.5.31 (Probability tightness does not imply runtime tightness due to stupid reasons). Let RWS be some normal rewinding strategy. Construct RWS′ from RWS by running for 2^λ steps and then emulate RWS. Clearly, RWS′ and RWS are equivalent systems, but runtime tightness of RWS′ is exponential.

Nevertheless, for typical classes of well-behaved protocols and rewinding strategies, runtime tightness, "query tightness", and probability tightness are closely related.

C.5.6. ⋆ Absolute Notions of Relative Efficiency

In Section 5.4.2, we work with "relative notions of (relative) efficiency", that is, we compare the performance of two algorithms. A scrapped approach used "absolute notions of relative efficiency", which have no comparison point. Absolute relative efficiency ensures, that whenever the communication partner of A is efficient, so is A. In other words, it allow us to "blame" a party for running too long. While this is easier to describe than relative efficiency, the need to be absolute makes the notion brittle, as we see at the end of this section.

We use the name *absolute relative efficiency* mostly due to a lack of a better name.

Definition C.5.32 (Weak absolute relative efficiency). Let $\mathcal{T}$ be a runtime class and A be an algorithm. Then A is **weakly absolutely relatively efficient (ar-eff)** (w.r.t. $\mathcal{T}$) if: For any timeful oracle $\mathcal{O}$, $\mathsf{time}_{\mathcal{O}}(\langle \mathsf{A}, \mathcal{O}\rangle) \in \mathcal{T}$ implies $\mathsf{time}_{\mathsf{A}+\mathcal{O}}(\langle \mathsf{A}, \mathcal{O}\rangle) \in \mathcal{T}$.

Definition C.5.33 (Absolute relative efficiency). Let A be an algorithm and $\mathcal{O}$ an oracle. Then A is **absolutely relatively efficient (ar-eff)** w.r.t. $\| \cdot \|_q$ with **rel-eff ratio** $\mathsf{poly}_{\mathsf{arr}}(\lambda)$ if: For any timeful oracle $\mathcal{O}$, we have $\|\mathsf{time}_{\mathsf{A}+\mathcal{O}}(\langle \mathsf{A}, \mathcal{O}\rangle)\|_q \leq \mathsf{poly}_{\mathsf{arr}}(\lambda) \cdot \|\mathsf{time}_{\mathcal{O}}(\langle \mathsf{A}, \mathcal{O}\rangle)\|_q$.

If q is not specified, we mean $q = \infty$, i.e. ar-eff w.r.t. to strict time. To $q = 1$, we say ar-eff **w.r.t. expectation**.

Importantly, the notion of (weak) absolute relative efficiency is *unconditional and amortized* since $\mathcal{O}$ is timeful and can abort at any time. In particular, if an algorithm is *PPT (resp. EPT) per activation*, it is (weakly) ar-eff.

As a rule of thumb, the *non-adversarial parties* (e.g. challengers) should be ar-eff, so that runtime problems can be traced back to the adversary. With this, one can exploit runtime explosions to break hardness assumptions.

Remark C.5.34 (Relation to EPT in any interaction). At fist glance, ar-eff (w.r.t. expectation) seems to be closely related to *EPT in any interaction (EPTiai)* [KL08; Gol10]. However, EPTiai has a different flavour. It is a property imposed on the (ideal) adversary, so that a simulator's runtime does not explode. Katz and Lindell state in[KL08, Sec. 4.2] that they could not show that the simulator obtained by modular sequential composition again satisfies EPTiai. This "prevents" further composition of this type. Conversely, ar-eff is a property imposed "honest parties", e.g. challenger in a security game.

Example C.5.35 (G3C is not ar-eff). The prover, verifier and simulator for $\mathsf{G3C}_{\mathsf{GK}}$ (Section 5.1.2) are ar-eff under size-guarding, but *not* unguarded, assuming $|(V, E)| \approx \mathsf{card}(V) + \mathsf{card}(E)$. The problem is that the P makes $\lambda \cdot \mathsf{card}(E) \cdot \mathsf{card}(V)$ commitments, whereas the verifier only makes $\mathsf{card}(E)$ commitments. The factor $\mathsf{card}(V)$ is not bounded by $\mathsf{poly}(\lambda)$, thus, there is no $\mathsf{poly}_{\mathsf{arr}}$ which depends only on λ and the prover is not ar-eff.[33]

This problem is mitigated by size-guards. For a variation of $\mathsf{G3C}_{\mathsf{GK}}$ with graph hamiltonicity, this problem would not occur as λ parallel repetitions suffice, independent of G. (Modulo technical complications.)

[33] This can be seen as a technical artefact from not counting the commitments sent by the prover towards the runtime of the verifier. An honest verifier would read the commitments, hence requiring roughly the same amount of "computation" as the prover. A dishonest or timeful verifier is not bound by that. If we would count incoming messages towards the runtime of the oracle, stupid problems like "message length doubling attacks" could appear. By sending a message m, A gets $\mathsf{poly}_{\mathsf{arr}}(\lambda) \cdot |m|$ more time from $\mathcal{O}$. If $\mathcal{O}$ discards messages which are too long, then $\mathcal{O}$ can remain efficient, whereas $\mathcal{A}$ increases its runtime exponentially. Arguably, we do not want to view such an A as efficient in any sense.

All in all, Example C.5.35 demonstrates how brittle *unguarded* ar-eff is. We see that not even the prover of $\mathrm{G3C_{GK}}$ satisfies ar-eff without size-guards. This is the core reason to replace ar-eff with the arguably more complex notion of "efficiency relative to" another algorithm.

C.5.7. ⋆ The Necessity of $\|\cdot\|_1$

One may hope that there is a notion more stringent than expected time, which still allows rewinding-based arguments of 3-move proofs of knowledge (based on special soundness), or 5-move zero-knowledge such as [GK96], with black-box proofs of security. For example, one might hope for $\|\cdot\|_2$ instead of $\|\cdot\|_1$, i.e. expected polynomial time and variation. Unfortunately, it is unlikely that a satisfying solution exists, at least along this line of arguments, unless one allows a larger (constant) number of rounds.

Concretely, consider the setting of 3-move proofs of knowledge. There, one can assume an adversary which plays honestly, but aborts with probability $1-p \in [0,1]$. Suppose the 3-move proof of knowledge is special sound and has large challenge space, so that it the soundness error is negligible. Consider the typical extractor for special soundness: If the adversarial prover convinces the verifier, it rewinds and uses honestly sampled challenges until the adversary produces a second convincing answer. With overwhelming probability, the first and second challenge are distinct, and by special soundness a witness can be computed.

It is evident that the number of rewinds for this extractor follows a geometric distribution. Indeed, with probability p, the first challenge is answered convincingly, in which case the extractor needs $R \sim \mathrm{Geo}(p)$ rewinds to obtains a second convincing answer. Then the expectation of R is

$$\|R\|_1 = (1-p)\cdot 0 + p\frac{1-p}{p} \leq 1.$$

If we consider $\|R\|_2$, then we find the

$$\|R\|_2^2 \geq p\frac{1-p}{p^2} \geq \frac{1}{p}.$$

Thus, if p negligible, the $\|R\|_2$ is superpolynomial. A first attempt is to exploit virtuality: If p is negligible, then in fact, R^2 is virtually expected polynomial. Conversely, if p is bounded below by $\frac{1}{\mathrm{poly}}$, then R^2 is expected polynomial. However, things fall apart if $p = p(\lambda)$ is not negligible, yet not bounded below by any polynomial. For this, define $p(\lambda)$ as follows: $p(\lambda) = \lambda^{-2^{f(\lambda)}}$, where $f(\lambda) = 1$ for all $2 \nmid \lambda$, $f(\lambda) = 2$ for all $2 \mid \lambda \wedge 4 \nmid \lambda$, $f(\lambda) = 3$ for all $4 \mid \lambda \wedge 8 \nmid \lambda$, and so on. (Let $f(0) = 0$.) That is,

$$(f(\lambda))_\lambda = (0,1,2,1,3,1,2,1,4,1,2,\ldots)$$

Thus, p contains infinitely many terms of λ^{-2^c} for any $c \in \mathbb{N}$. It is easy to see that p is not negligible. However, for any polynomial poly, we have $p < \frac{1}{\mathrm{poly}}$ infinitely often, i.e. p is not polynomially bounded away from 0. Thus, $\|R\|_2 > \mathrm{poly}$ infinitely often. In other words, there is no polynomial which bounds $\|R\|_2$. Allowing negligible virtuality does not help either. Thus, this choice of p results in an adversary which cannot be extracted in virtually expected polynomial $\|\cdot\|_2$-time.

Repetitions can be used to "bring down exponents", and hence, for any $q \in \mathbb{N}$, there should exist a (constant) number C of repetitions (and modified extractors) such that $\|R\|_q < \mathrm{poly}$, namely $C = q$. This may be interpreted as an intermediate result between proofs of knowledge with EPT extraction (i.e. $q = 1$), and "strong proofs of knowledge" with PPT extraction (i.e. $q = \infty$). We also refer to [Pas06] where this is discussed in the context of precise zero knowledge proofs (of knowledge).

C.5.8. ⋆ Measurability

In this section, we discuss questions of measurability, which we ignored elsewhere. Since all of our constructions are simple and make no use of the axiom of choice, there is little reason to doubt that all are measurable. Admittedly, we have not formally verified this for every construction, and merely spot-checked some. We do note that some properties, e.g. "uniqueness" of events, were used in simplified explanations. They are not used in actual constructions.

Stochastic Processes and Timeful Systems. The evolution of a computation or interaction for closed systems should be viewed as a stochastic process. The random variables of interest are the exchanged messages, and the purported elapsed time,[34] namely the sequence of random variables $(Z_0, Z_1, \dots)$ describing the progress of the computation. Concretely, Z_i consists of $(m_0, t_0, \dots, m_i, t_i)$, which is transcript up to the i-th message exchange, plus the elapsed runtime t_j for computing m_j. One may augment this with other (purported) values, such as memory usage, etc. Obviously, we also require $\mathrm{pr}[1, \dots, j](Z_i) = Z_j$ for all $i < j$ in $\mathbb{N}_0$. (And this implies $\sigma(Z_i) \subseteq \sigma(Z_{i+1})$ for the σ-algebras.)

The image of Z_i lies is in a countable space, which we equip with the discrete σ-algebra. The sample space of process $Z(i, \omega) = Z_i(\omega)$ is given the induced σ-algebra, but is not countable anymore. (Recall that the σ-algebra on $\mathbb{N}^\infty$ is constructed from the finite steps $\mathbb{N}^k$, $k \in \mathbb{N}$.)

We have ignored inputs and non-closed systems, but these are easily defined as *functions*, which take a sequence of input messages and return output messages. (Technically, these may be partial functions, since some input sequences may correspond to impossible executions. E.g. inputs for a system which halted.) Letting two such systems A, B interact by connecting interfaces yields a new system, defined in the obvious way. The resulting system $\langle \mathsf{A}, \mathsf{B} \rangle$ has an associated random process (which lives in the product space of the random processes associated with A, B.)

An Alternative Description. One may alternatively describe the random process of individual systems via "conditional transition probabilities", roughly, $p_i(\vec{y}) = \Pr[Z_i = \vec{y} \mid Z_{i-1} = \mathrm{pr}[1, \dots, i-1](\vec{y})]$. This approach always specifies independent processes. Dependency is (only) introduced by interaction. While this would probably be sufficient, "extending" the probability space (as we did to achieve exact ν-quantile cutoffs) is not immediately possible. One can introduce some "irrelevant action", e.g. a zero-th message, which has the desired distribution. We find this to be just as inconvenient as working with underlying probability spaces explicitly. Moreover, for systems induced by algorithms and machine models, the underlying probability space is usually explicit anyway.

Algorithms and Machine Models. Unlike (timeful) systems, algorithms and machine models have an "explicit randomness-providing interface". Thus, the underlying probability space for such systems is simple to describe, usually $\{0,1\}^{\mathbb{N}}$ with "uniform" distribution (i.e. the "limit" of $\{0,1\}^k$, $k \in \mathbb{N}$, with uniform distribution). Standard definitions of machine models (via transition functions) then evidently imply that any algorithm yields systems which are very well behaved, in particular every typical function of interest is measurable (e.g. messages, runtime, memory trace, …).

[34] In particular, a *timeful* system must have measurable purported runtime.

Measurability of our Constructions. Most constructions merely relied on runtime statistics, and can be defined on the process Z_i by (a consistent family of) measurable functions (for each i). Indeed, if the domain of Z_i has the discrete σ-algebra, any function is measurable. Since these statistics are measurable by assumption, and our functions are measurable as well, we therefore find that our constructions are measurable. (More concretely, they are measurable for every i, and hence the resulting process is measurable.)

C.6. ⋆ Extendability from Indistinguishable Queries

Our definition of benign simulators relies on structure of the proof of security for (PPT) simulation, and, although it covers many examples, is therefore somewhat limited. In this section, we give a different approach to benign simulation. Intuitively, we require that an "eavesdropping" environment cannot distinguish the bb-rw interaction of a rewinding strategy or a simulator with V^*. This corresponds to the properties of query-indistinguishability and zero-knowledge.

The upside of this approach is its apparent greater generality. The downside is, that using query-indistinguishability is more technical, and requires a separate treatment of efficiency and indistinguishability. Perhaps a better, general approach exists — yet we know none.

C.6.1. Query-Sequences Indistinguishability

Our notion of "indistinguishable queries" for simulators is similar in spirit to [KL08].

Definition C.6.1 (Query-indistinguishability). Let A and B be oracle algorithms. The distinguishing advantage $\mathrm{Adv}^{\mathrm{qseq}}_{(\mathcal{I},\mathcal{O},\mathcal{D}),\mathsf{A},\mathsf{B}}(\lambda)$ for queries *including output* of A, B by an adversary $(\mathcal{I},\mathcal{O},\mathcal{D})$ is defined as the distinguishing advantage $\mathrm{Adv}^{\mathrm{dist}}_{\mathcal{D},X,Y}(\lambda)$ for the distributions

$$X := \{(x, y, r, \mathsf{A}^{\mathcal{O}(y;r)}(x; r_{\mathsf{A}}), \mathsf{qseq}_{\mathcal{O}}(\mathsf{A}^{\mathcal{O}(y;r)}(x; r_{\mathsf{A}}))) \mid (x, y) \leftarrow \mathcal{I}(\lambda)\}_\lambda$$
$$Y := \{(x, y, r, \mathsf{B}^{\mathcal{O}(y;r)}(x; r_{\mathsf{B}}), \mathsf{qseq}_{\mathcal{O}}(\mathsf{B}^{\mathcal{O}(y;r)}(x; r_{\mathsf{B}}))) \mid (x, y) \leftarrow \mathcal{I}(\lambda)\}_\lambda$$

Here r denotes the *accessed* randomness of $\mathcal{O}$.[35] (We make explicit the randomness r_{A} and r_{B} only to make it evident, that the output and query sequence refer to the same run.)

We say that A and B satisfy **($\mathcal{T}$-time) query-indistinguishability (Q-IND)**, if for all adversaries $(\mathcal{I},\mathcal{O},\mathcal{D})$ such that $\mathsf{time}_{\mathcal{I}+\mathcal{O}+\mathcal{D}}(\mathsf{A}^{\mathcal{O}}) \in \mathcal{T}$ and $\mathsf{time}_{\mathcal{I}+\mathcal{O}+\mathcal{D}}(\mathsf{B}^{\mathcal{O}}) \in \mathcal{T}$ the advantage $\mathrm{Adv}^{\mathrm{qseq}}_{(\mathcal{I},\mathcal{O},\mathcal{D}),\mathsf{A},\mathsf{B}}(\lambda)$ is negligible.

Size-guarded query-indistinguishability is defined by size-guarding A and B (as non-adversarial parties), i.e. A and B reject inputs of length larger than their size-guard.

Definition C.6.1 requires *jointly* indistinguishable *queries* and *outputs.* This may not be strictly necessary, but greatly simplifies sequential composition of Q-IND. All bb-rw zero-knowledge simulators we are aware of satisfy this joint indistinguishability. Indeed, typically the last query induces the (purported) view of the adversary.

[35] Typical machine models offer an infinite pool of (independent) randomness, e.g. a random tape. Thus, we "restrict" to accessed randomness.

We stress that the distinguisher $\mathcal{D}$ learns the oracle randomness. This allows $\mathcal{D}$ to replay the execution of $\mathcal{O}$, recover the complete transcript of the execution, and compute the runtime spent in $\mathcal{O}$.

Remark C.6.2. For CEPT and CPPT, it suffices in Definition C.6.1 to require that $T = \mathsf{time}_{\mathcal{I}+\mathcal{O}+\mathcal{D}}(\mathsf{A}^{\mathcal{O}}) \in \mathcal{T}$. If $\mathsf{time}_{\mathcal{I}+\mathcal{O}+\mathcal{D}}(\mathsf{B}^{\mathcal{O}}) \notin \mathcal{T}$, then this is a distinguishing statistic. Indeed, by a standard reduction to PPT, any distinguisher $(\mathcal{I}, \mathcal{O}, \mathcal{D})$ with advantage at least $\varepsilon = \mathsf{poly}^{-1}$ (infinitely often) can be truncated to an *a priori* PPT distinguisher with advantage $\frac{\varepsilon}{4}$ (infinitely often). (Just interpret $\mathcal{D}' \mathrel{\widehat{=}} (\mathcal{I}, \mathcal{O}, \mathcal{D})$ as interacting with oracles A or B, and apply Corollary 5.4.2.)

Remark C.6.3 ("Universal" adversary, environments, sequential security). Using a universal machine for $\mathcal{O}$ (and even $\mathcal{D}$) gives a universal adversary similar to zero-knowledge. Moreover, one can rephrase Definition C.6.1 in terms of an "environment" $\mathcal{E}$ which encompasses $\mathcal{I}$ and $\mathcal{D}$; $\mathcal{E}$ sends inputs (x, y), and then gets access to randomness, output and query sequence. A sequential security version of Q-IND is defined by this approach, following the definition of sequential zero-knowledge (i.e. $\mathcal{E}$ is given adaptive repeated trials).

As in Remark C.6.2, we may assume $\mathcal{E}$ is a priori PPT. Also, (one-guess) "environmental" security is equivalent to Definition C.6.1, because the state of $\mathcal{E}$ can be encoded as part of y.

C.6.2. Adapting the Result of Katz–Lindell

As a warm-up, we adapt the result of Katz and Lindell [KL08]. For that, we rely on bb-rw simulators which are EPT for *any* adversary (not counting the adversary's steps). In other words, we rely on simulators which are normal in the sense of Goldreich [Gol10]. That covers most simulators in the literature, but not our naive simulator for $\mathsf{G3C}_{\mathsf{GK}}$. Moreover, the definition is not compatible with expected polynomial input sizes, and thus restricted to size-guarded security.[36] (For size-guards, see Appendix C.5.4 and Appendix C.5.4.3) After this motivation, we generalize the result to our setting.

Definition C.6.4 (Goldreich-normal [Gol10]). A bb-rw simulator is **normal in the sense of Goldreich**, short **Goldreich-normal**, if for any (not necessarily computable) timeful V^* and any input $(\mathbb{x}, \mathbb{w}, aux)$ (with $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$) there is a polynomial poly such that $\mathbb{E}[\mathsf{time}_{\mathsf{Sim}}(\mathsf{Sim}(\mathbb{x}, \mathsf{V}^*(aux)))] \leq \mathsf{poly}_{\mathsf{Sim}}(|\mathbb{x}|)$,

There is no requirement of $\mathbb{x} \in \mathcal{L}$ in [Gol10, Definition 6]. Since zero-knowledge only quantifies over such statements, we have adapted the definition to fit.

Lemma C.6.5 (Auxiliary input zero-knowledge). *Let* (P, V) *be an argument system. Let* Sim *be a (timed) bb-rw simulator with associated rewinding strategy* RWS. *Suppose that* RWS *is normal,* Sim *is Goldreich-normal,* RWS *and* Sim *have indistinguishable queries, and* Sim *handles PPT adversaries in EPT.*

Then Sim *handles CEPT adversaries in CEPT* under size-guarding, *and* (P, V) *is size-guarded zero-knowledge.*

Proof sketch. By a standard reduction, the output quality of Sim can be tested by an a priori PPT adversary. By assumption, such output is indistinguishable from the real protocol. Thus, we only need to ensure efficiency of Sim under size-guarding.

Due to size-guarding, we can assume that $\mathcal{I}$ outputs $\mathbb{x}$ with $|\mathbb{x}| \leq \mathsf{poly}_{\mathcal{I}}(\lambda)$. Therefore, by assumption, Sim is *a priori* EPT with bound $\mathsf{poly}_{\mathsf{Sim}}(\mathsf{poly}_{\mathcal{I}}(\lambda))$, *excluding* the time spent in the bb-rw oracle V^*. Thus,

[36] These problems do no surface in [KL08; Gol10] since they define $\lambda = |\mathbb{x}|$.

it is sufficient to bound $S_{\mathsf{V}^*} = \mathsf{time}_{\mathsf{V}^*}(\mathsf{Sim}(\mathbb{x}, \mathsf{V}^*(aux)))$, where $(\mathbb{x}, \mathbb{w}, aux) \leftarrow \mathcal{I}$. By normality of RWS and query-indistinguishability, recomputing the time spent in V^* by emulation (using inputs, queries and randomness) is possible in CEPT for any CEPT adversary. Consequently, by query-indistinguishability, switching from RWS to Sim results in an indistinguishable distribution of t. Hence, S_{V^*} is CEPT and the claim follows. □

We also demonstrate that sequential composition, i.e. sequential zero-knowledge, holds for this type of simulator.

Lemma C.6.6 (Sequential zero-knowledge)**.** *Let* (P, V) *be a size-guarded zero-knowledge argument, with a simulator satisfying the conditions in Lemma C.6.5. Then* (P, V) *is sequential size-guarded zero-knowledge.*

Proof sketch. Again, the main question is efficiency. Namely, if there is a distinguishing adversary for zero-knowledge, then there is an a priori PPT adversary. This contradicts our assumptions, because "classical" sequential composition against a priori PPT adversaries holds.

To prove efficiency, we prove, essentially, that query-indistinguishability composes sequentially. As in Lemma C.6.5, this then implies that $\mathcal{O}_{\mathsf{Sim}}$ is efficient because $\mathcal{O}_{\mathsf{RWS}}$ is.

Suppose the contrary, i.e. suppose query-indistinguishability does not hold for Sim. By Remarks C.6.2 and C.6.3, we know that there is an a priori PPT distinguisher $(\mathcal{E}, \mathsf{V}^*)$ breaking "sequential query-indistinguishability". We leave the definition of sequential Q-IND, sketched in Remark C.6.3, to the reader.

Since Sim is Goldreich-normal, $\mathcal{O}_{\mathsf{Sim}} = \mathsf{rep}(\mathsf{Sim}(\cdot))$ handles PPT adversaries in EPT. (This is "classical" sequential composition.) Sequential Q-IND of Sim and RWS for PPT distinguishers reduces, by a hybrid argument, to standard Q-IND. The hybrid distinguisher is efficient, because Sim is Goldreich-normal (and RWS normal). Consequently, Sim and RWS cannot be Q-IND. A contradiction. Hence, sequential Q-IND holds for RWS and Sim. In particular, $\mathsf{rep}(\mathsf{Sim})$ satisfies all conditions in Lemma C.6.5 lifted to the sequential setting, and the proof lifts as well. □

This warm-up demonstrates two things: First, with size-guarding, many arguments get simplified and reduce to standard a priori PPT arguments. Second, the main difficulty for relaxations will be to demonstrate efficiency. By the nature of CEPT, efficiency and indistinguishability are somewhat entangled. We use unconditional guarantees, similar to Goldreich-normal in the above, to partially disentangle that.

Proving a "full-fledged" CEPT simulation, i.e. getting rid of size-guarding and weakening Goldreich-normal is surprisingly cumbersome. We do so by introducing two properties. The first property, *runtime estimators*, allows us to link together the runtime of RWS and Sim, *assuming Q-IND holds.* The second property ensures efficiency if one truncates after polynomially many queries. This replaces Goldreich-normal, and enables the hybrid argument which shows that Q-IND must hold under sequential composition. Taken together, we find that the runtime of Sim cannot be too far from RWS, and thus Sim is efficient whenever RWS is. This generalizes the proof of Lemma C.6.6.

C.6.3. Runtime Estimation

In the following, we give a definition of a “runtime estimator”, which allows to lower- and upper-bound the expected runtime of an algorithm depending on oracle queries, or more precisely, on the information available to a Q-IND adversary. The algorithms of interest are RWS and Sim. Typically, their runtime is closely related, since both emulate the honest prover (with minor modifications). Consequently, their runtime per activation is easy to lower- and upper-bound (if the prover’s runtime per activation is).

Definition C.6.7 (Runtime estimation). Let $\theta\colon \mathbb{N}_0 \times D \times \Omega_\theta \to \mathbb{N}_0$, be a probabilistic algorithm with randomness space Ω_θ, and where D is the input space of a query distinguisher (as in Definition C.6.1). Let A be an algorithm and $\mathcal{O}$ some oracle. Let $z \in D$ and recall that $z = (x, y, r, out, qs)$, where x (resp. y) is input to A (resp. $\mathcal{O}$), r is the oracle randomness, out is the output of $\mathsf{A}^{\mathcal{O}(y;r)}(x)$, and qs is the sequence of queries. Define

- $t_\theta(\lambda, z) := \mathbb{E}[\mathsf{time}_\theta(\theta(\lambda, z))]$, the expected runtime of θ given z.
- $t_{\mathsf{A}}(\lambda, z) := \mathbb{E}[\mathsf{time}_{\mathsf{A}}(\mathsf{A}^{\mathcal{O}(y;r)}(x)) \mid \mathsf{A}^{\mathcal{O}(y;r)}(x) = out \land \mathsf{qseq}_{\mathcal{O}}(\mathsf{A}^{\mathcal{O}(y;r)}(x)) = qs]$, the expected runtime of A conditioned on z.
- $t_{\mathsf{A}+\mathcal{O}}(\lambda, z)$ like t_{A}, but using $\mathsf{time}_{\mathsf{A}+\mathcal{O}}(\ldots)$.

We say that θ is a **runtime estimator** if it satisfies **efficiency**, i.e. there exists some $\mathsf{poly}(\lambda)$ such that for all $z \in D$ and all λ: $t_\theta(\lambda, z) \le \mathsf{poly}(\lambda) \cdot t_{\mathsf{A}+\mathcal{O}}(\lambda, z)$. Moreover, θ is a **lower bound estimator** if there exists some poly such that $\mathbb{E}[\theta(\lambda, z)] \le \mathsf{poly}(\lambda) \cdot t_{\mathsf{A}}(\lambda, z)$ for all $z \in D$ and $\lambda \in \mathbb{N}_0$. Analogously, θ is a **upper bound estimator** if there exists some poly such that $t_{\mathsf{A}}(\lambda, z) \le \mathsf{poly}(\lambda) \cdot \mathbb{E}[\theta(\lambda, z)]$ for all $z \in D$ and $\lambda \in \mathbb{N}_0$.

Note that estimators are “unconditional” constructions; we quantify over all $z \in D$.

Remark C.6.8 (Sketched application of runtime estimators). Consider a simulator Sim and its rewinding strategy RWS. If $T = \mathsf{time}_{\mathsf{RWS}}(\mathsf{RWS}^{\mathsf{V}^*})$ is CEPT, then the (output of the) runtime estimate θ is CEPT if it lower-bounds T. If θ upper-bounds $S = \mathsf{time}_{\mathsf{Sim}}(\mathsf{RWS}^{\mathsf{V}^*})$, then S is CEPT if (the output of) θ is. Since θ only depends on the information available to a Q-IND adversary, assuming RWS and Sim are Q-IND, the runtime bound provided by θ only changes negligibly, hence if T is CEPT, so is S. This provides a central link between the runtime RWS and Sim.

Remark C.6.9 (Convenience of size-guards). Arguing via runtime estimates requires that the algorithms runtime per activation behave somewhat regularly (which is fortunately typical). Most convenient are “essentially constant-time” algorithms (where runtime only depends on query/message length). With size-guards this is usually immediate, as every round has an a priori polynomial upper bound for the (expected) number of steps taken, both in RWS and Sim (not counting the black-box V^*). Hence θ is as simple as the total number of queries. Without size-guards, the behaviour is more fickle.

C.6.4. Efficiency from Query-Truncation

We already saw in Lemmas C.6.5 and C.6.6 that Q-IND ensures that the time spent in V^* only changes negligibly between RWS and Sim. However, we cannot reuse the arguments to show that Q-IND composes sequentially. The problem lies within efficiency of the hybrid distinguisher. As seen in Lemma C.6.6, once we obtain an a priori setting, this problem “disappears”. Hence this is our solution. We define what it means to be “Goldreich-normal for any polynomial query cutoff” of the interaction. Intuitively, it means that any “polynomial prefix” of the interaction is Goldreich-normal.

Definition C.6.10. Let A be an oracle-algorithm. Let $\mathsf{B}^{\mathcal{O}}(x, q)$ be the oracle-algorithm which emulates $\mathsf{A}^{\mathcal{O}}(x)$ until the q-th query of A to $\mathcal{O}$. After that, B returns `timeout` (otherwise B returns whatever A returns). We say A is **Goldreich-normal for any polynomial query cutoff** (and input space $\mathcal{X}_\lambda$), if for any polynomial poly_0 there is a polynomial poly_1, such that for any oracle $\mathcal{O}$, and any inputs $(x, y) \in \mathcal{X}_\lambda$ $\mathbb{E}[\mathsf{time}_\mathsf{B}(\mathsf{B}^{\mathcal{O}(y)}(x, \mathsf{poly}_0(\lambda)))] \leq \mathsf{poly}_1(|x|, \lambda)$. In other words, $\mathsf{B}(\cdot, \mathsf{poly}_0)$ is Goldreich-normal for any poly_0. For zero-knowledge, the input space is $\mathcal{R} \times \{0,1\}^*$.

Example C.6.11. Our rewinding strategy and simulator of $\mathrm{G3C}_{\mathrm{GK}}$ are Goldreich-normal for any polynomial query cutoff. Indeed, they are even PPT for any polynomial query cutoff. As a matter of fact, we cannot point out any (natural) bb-rw simulator which does not satisfy this property.

Remark C.6.12 (Goldreich-normal for any polynomial query cutoff does not imply efficiency). Similarly to size-guarding, restricting to a polynomial number of queries makes simulations efficient which would otherwise not be. For example, a simulator which is a a priori PPT per activation, but never halts, is Goldreich-normal for any polynomial query cutoff.

C.6.5. Query-Benign Simulators

Now, we bring together our definitions to define an alternative of benign, which we call query-benign.

Definition C.6.13 (Query-benign simulator). Let (P, V) be an argument system. Let Sim be a *(timed) bb-rw* simulator with **associated rewinding strategy** RWS. Then Sim is **query-benign** if

1. RWS is a *normal* and has a runtime estimator θ;
2. for all a priori PPT adversaries $(\mathcal{I}, \mathsf{V}^*)$, RWS and Sim satisfy Q-IND;
3. Sim is Goldreich-normal for any polynomial query cutoff.

Query-Benign under size-guard gd is as usual (i.e. by query-benign w.r.t. the size-guarded prover).

Recall that Q-IND (i.e. condition item 2) implies that RWS and Sim have indistinguishable outputs by definition, i.e Q-IND implies zero-knowledge. In 2 we use a priori PPT adversaries, since the security is equivalent to CEPT anyway.

Now, we put our definitions to use. Since our arguments are very close to Lemma C.6.6, we directly show sequential zero-knowledge.

Lemma C.6.14 (Query-benign implies sequential zero-knowledge)**.** *Suppose* (P, V) *is an argument system. Let* Sim *be a query-benign simulator. Then* (P, V) *is sequential zero-knowledge. In particular,* Sim *handles CEPT adversaries in CEPT. The analogous claim holds under size-guarding.*

Our proof is only a sketch and somewhat hand-wavy. in particular, we leave sequential security definitions, like "sequential Q-IND" and "sequential runtime estimators", and many straightforward arguments to the reader.

Proof sketch. As usual, the proof consists of two parts. First, we prove that using Sim instead of P is still CEPT. Then, by standard arguments, zero-knowledge follows. For simplicity, we argue assuming the real execution halts with probability 1.

Step 1 (Replacing RWS): As in Lemma 5.6.26, using $\mathsf{rep}(\mathsf{RWS}(\,\cdot\,))$ instead of $\mathsf{rep}(\langle \mathsf{P}, \cdot \rangle)$ in the sequential zero-knowledge experiment is still CEPT and the output distribution is unchanged.[37]

Step 2 (Goldreich-normal for any polynomial query cutoff composes sequentially): It is straightforward to verify that if an algorithm B is Goldreich-normal for any polynomial query cutoff, so is its "repetition" $\mathsf{rep}(\mathsf{B})$. For this compare, the poly-query truncation of $\mathsf{rep}(\mathsf{B})$ with $\mathsf{rep}(\mathsf{B}_0)$, where B_0 is the poly-query truncation of B. Since B_0 is Goldreich-normal, so is $\mathsf{rep}(\mathsf{B}_0)$. Consequently, $\mathsf{rep}(\mathsf{B})$ is Goldreich-normal for any polynomial truncation.

Step 3 (Q-IND holds for $\mathsf{rep}(\mathsf{RWS})$ and $\mathsf{rep}(\mathsf{Sim})$): Now, consider the "sequential Q-IND" experiment, i.e. consider Q-IND of $\mathsf{rep}(\mathsf{RWS})$ and $\mathsf{rep}(\mathsf{Sim})$. More concretely, the distinguishing environment $\mathcal{E}$ that can repeatedly invoke $\mathsf{RWS}^{\mathsf{V}^*}$ resp. $\mathsf{Sim}^{\mathsf{V}^*}$, and obtains the output of an invocation, including the query sequence and randomness of (that invocation of) V^*, as noted in Remark C.6.3. Note that $\mathcal{E}$ can adaptively choose inputs to RWS resp. Sim and V^*.

W.l.o.g., we may assume that $\mathcal{E}$ is a priori PPT, say $\mathcal{E}$ makes at most $\mathrm{poly}_{\mathcal{E}}$ steps. Moreover, we may assume that $\mathcal{E}$ *linearly reads* the outputs of each invocation. In particular, $\mathcal{E}$ cannot skip (parts) of the outputs, and read only the final queries.[38] Importantly, $\mathcal{E}$ only reads a strict polynomial prefix of the full (sequential) query sequence.

If we replace Sim by a truncation Sim_0, which stops after $\mathrm{poly}_{\mathcal{E}}$ queries, we know that Sim_0 is EPT with expected runtime bounded by some $\mathrm{poly}_{\mathsf{Sim}_0}$ (due to Sim being Goldreich-normal for any polynomial query truncation, see also Step 2).

By construction, from the perspective of $\mathcal{E}$, $\mathsf{rep}(\mathsf{Sim}_0)$ and $\mathsf{rep}(\mathsf{Sim})$ behave identically. Indeed, since $\mathcal{E}$ only reads at most a prefix of length $\mathrm{poly}_{\mathcal{E}}$ of the (total) query sequence, $\mathcal{E}$ never encounters the difference of Sim_0 and Sim. For symmetry, let RWS_0 be defined analogously to Sim_0. (Formally, we could use RWS, since there are no efficiency problems with RWS.)

Now, we can use that Sim_0 is Goldreich-normal, to show via a hybrid argument as in Lemma C.6.6 that if $\mathcal{E}$ can distinguish RWS_0 and Sim_0 for "sequential Q-IND", there is a Q-IND distinguisher $\mathcal{D}$ for RWS_0 and Sim_0. And hence, there is a Q-IND distinguisher for RWS and Sim (since the constructed hybrid distinguisher $\mathcal{D}$ also sees no difference between Sim_0 and Sim (resp. RWS_0 and RWS)). Thus, "sequential Q-IND" holds.

Step 4 (Sim is CEPT if RWS is): Now we make use of the runtime estimator θ. More precisely, we extend θ to the sequential setting by applying the underlying θ for each invocation separately, and taking the sum of the estimates. It is easy to see that this preserves efficiency, lower-bounding and upper-bounding.

Let $(\mathcal{E}, \mathsf{V}^*)$ be a CEPT adversary. Since $\mathrm{time}_{\mathsf{RWS}}(\langle \mathcal{E}, \mathsf{rep}(\mathsf{RWS}^{\mathsf{V}^*})\rangle)$ is CEPT, so is θ (by lower-bounding of $\mathsf{RWS}^{\mathsf{V}^*}$). Since $\theta(z_{\mathsf{RWS}})$ is CEPT for $z_{\mathsf{RWS}} = \mathsf{qseq}_{\mathsf{V}^*}(\mathsf{rep}(\mathsf{RWS}^{\mathsf{V}^*}))$ and since z_{RWS} and $z_{\mathsf{Sim}} =$

[37] In case of non-halting executions, argue as in Lemma 5.6.23.

[38] This is a technical requirement. Depending on the machine model, $\mathcal{E}$ may have random access to the output. That would make our later argument incomplete. To see that we can assume that $\mathcal{E}$ completely reads the outputs, just use the output length as a distinguishing statistic. That is, if there is a PPT distinguisher $\mathcal{E}$ which skips parts of the output, then the variation which reads *all of the output* is still CEPT for RWS. By standard truncation arguments, an a priori PPT truncation of $\mathcal{E}'$ retains non-negligible advantage. And $\mathcal{E}'$ is a distinguisher of the kind we are interested in.

$\mathsf{qseq}_{\mathsf{V}^*}(\mathsf{rep}(\mathsf{Sim}^{\mathsf{V}^*}))$ are indistinguishable w.r.t. $\mathcal{E}$ (by "sequential Q-IND"), also $\theta(z_{\mathsf{Sim}})$ is CEPT. Since θ upper-bounds the runtime of $\mathsf{time}_{\mathsf{Sim}}(\langle\mathcal{E}, \mathsf{rep}(\mathsf{Sim}^{\mathsf{V}^*})\rangle)$, $\mathsf{time}_{\mathsf{Sim}}(\langle\mathcal{E}, \mathsf{rep}(\mathsf{Sim}^{\mathsf{V}^*})\rangle)$ is CEPT.

Finally, since the time spent in V^* can be easily reconstructed from z (by emulating the execution), $\mathsf{time}_{\mathsf{V}^*}(\langle\mathcal{E}, \mathsf{rep}(\mathsf{RWS}^{\mathsf{V}^*})\rangle) \overset{c}{\approx} \mathsf{time}_{\mathsf{V}^*}(\langle\mathcal{E}, \mathsf{rep}(\mathsf{Sim}^{\mathsf{V}^*})\rangle)$ due to Q-IND.

All in all, replacing $\mathsf{rep}(\langle\mathsf{P}, \cdot\rangle)$ with $\mathsf{rep}(\mathsf{Sim}(\cdot))$ preserves CEPT.

Step 5 (Output quality): Our definition of Q-IND included the outputs, so zero-knowledge follows. □

The proof sketch should be interpreted as follows: Step 3 shows that Q-IND for A and B composes sequentially if B is Goldreich-normal for any polynomial query cutoff. (It uses Step 2, although somewhat indirectly.) Step 4 shows that runtime estimators compose sequentially. Taken together, query-benign composes sequentially. Lastly, (sequential) query-benign implies (sequential) zero-knowledge.

We remark that to prove Q-IND, for all of our examples, one essentially proves benigness as well.