Karlsruhe Institute of Technology

On Improving Communication Complexity in Cryptography

zur Erlangung des akademischen Grades einer

Doktorin der Naturwissenschaften

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

genehmigte

Dissertation

von

Lisa Maria Kohl

aus München

Tag der mündlichen Prüfung: 22. Oktober 2019

1. Referent: Prof. Dr. Dennis Hofheinz
2. Referentin: Prof. Dr. Elette Boyle

Acknowledgments

First and foremost, I would like to thank my advisor Dennis Hofheinz, who was the most supportive advisor one could wish for. He gave me the confidence to ask any question and encouraged me to bother him even with half-baked ideas he showed impossible within seconds, thereby keeping me from wasting my time. He would take an afternoon to explain me a paper, just to save me the time to read it. Thank you, Dennis, for everything you taught me (in particular, your high-level view of things), and for giving me all the freedom, support and help that made my PhD a through-and-through joyful experience.

Secondly, I want to thank Elette Boyle, who greatly influenced me and who I see as a role model. She is so positive and fun and welcoming that working with her was easy from the very first moment. She would dedicate her time to day-long research meetings, resulting in the most productive days of my life. She would always take care to meet me where I was, when I was lost in a current discussion. Thank you, Elette, for inviting me to Israel with open arms, for giving me a new perspective on research and cryptography, for showing me that coffee shops are the best places to work at, and for your continuing support and encouragement.

Next, I want to thank all my other co-authors, which I count myself very lucky to have worked with. Romain Gay, who taught me to always look at the simplest solution first. Jiaxin Pan, who is very caring both towards plants and people (thanks, Jiaxin, for guiding us through Beijing restaurants and sorry for still not being able to pronounce your name correctly). Julia Hesse, who always asks the right questions, who I love sharing offices and apartments with, and whom I'm deeply thankful to for continuing professional and personal advice. Peter Scholl, who seems to be able to magically fix every research problem within a day. Niv Gilboa, who always speaks precisely, resulting in very productive research discussions. Yuval Ishai, who is incredibly friendly and dedicated, who there is so much to learn from (of which I unfortunately know so little yet) and who I am very thankful to for the opportunity to continue my research with as a postdoc. Geoffroy Couteau, who I admire for his dedication to science in general, who makes me view the world in a different way with every conversation, and who is very supportive both professionally and personally. And Peter Rindal, who I envy for his programming superpowers.

I also want to thank my hopefully to-be co-authors, with whom I have (more or less) ongoing research projects: Alon Rosen, who always kindly shares his wisdom about research and life in general and who I hope to have many more opportunities to learn from. Marshall Ball, who I find highly inspiring and always very much enjoy spending time with, in particular for research and food (thank you, Marshall, for the productive and fun research week and unforgettable food tour in New York City). And Eylon Yogev, who is great to work with in particular for being very determined

in solving problems.

I also would like to thank every one of my colleagues at KIT for making me look forward to going to office every day, I could not imagine a friendlier working atmosphere. In particular, I want to thank Jessi for always listening, Thomas for always being supportive and Bogdan for always helping.

And, of course, I would like to thank every one I met during my wonderful research visit in Israel, in particular my fellow visitors, who made even the one-hour bus drive to IDC worth it.

I would like to thank Ronald Cramer, from whom I learned so much, for supervising my Master's thesis at CWI Amsterdam and for thereby taking great part in manifesting the wish in me to start a research career in the field of cryptography. And I would like to thank every one in the cryptology research group at the time for making it such a fun experience.

In addition, I would like to thank every one in the crypto community for creating the warm, friendly and open atmosphere that makes me look forward to every conference, workshop and school. I met so many amazing people over the years, many of which I'm happy to call friends now. I hope I will always be part of this community, which already now feels like a big family. In particular, I would like to thank all organizers of schools and workshops that I have profited from so greatly over the years: The Workshop on Theory and Practice of Secure Multi-Party Computation in Aarhus 2016 and in Bristol 2017, CrossFyre 2016, 2017 and 2019, the School on Randomness in Barcelona 2016, the BIU Winter School 2018 and 2019, the Crypto in the Galilee Workshop 2018, the Women in Theory Workshop 2018, the Beyond Crypto Workshop 2018, the Bertinoro Workshop on Lower Bounds 2019 and the New Roads to Cryptopia Workshop 2019.

Going further back in time, I would like to thank my math teacher Marianne Gerny for her enthusiasm and her dedication to teach us way beyond what would have been required, which greatly influenced me in my choice of studies and therefore everything that followed.

I would like to thank the advisor of my Bachelor's thesis Stefan Kühnlein, for his great support and encouragement and for introducing me to the beautiful world that is algebra. I also want to thank every one who made my studies such a joyful time, both in and outside university at KIT and during the months I spent at NUS.

I would like to thank Derek Dreyer for taking me as an intern at MPI-SWS in Saarbrücken, for showing me what research really is and for enabling me to go to a conference for the first time. I also want to thank the PhD students who were there at the time, for their support and the fun times I had.

Finally, I would like to thank my family and friends, who would bear me working during vacations, patiently listen to my practice talks and always support me. There is no way I would have made all this without their backing. I want to thank my parents for always believing in me without putting any pressure or expectations on me. I want to thank my brother for being the best brother imaginable and for his incredible support. I want to thank my grandfather Wolfgang and my uncle Max for showing true interest in my work and many interesting conversations. I want to thank my best friend Lena for the fact that I can always count on her one hundred percent. I want to thank Pia for distracting me when needed, I want to thank Lena for encouraging me, and I want to thank Sarah for being there for me. I also want to thank every one else I am lucky to call a friend for the support and for making

me the person I am today.

Unfortunately, time and space is not sufficient for giving every one the credit that is deserved and that I would like to give. But when reading this, if you were part of this journey you can be sure I thought of you at some point writing these acknowledgments, and I also very much want to thank you.

Contents

Abstract

Cryptography grew to be much more than "the study of secret writing". Modern cryptography is concerned with establishing properties such as privacy, integrity and authenticity in protocols for secure communication and computation. This comes at a price: Cryptographic tools usually introduce an overhead, both in terms of communication complexity (that is, number and size of messages transmitted) and computational efficiency (that is, time and memory required). As in many settings communication between the parties involved is the bottleneck, this thesis is concerned with improving communication complexity in cryptographic protocols.

One direction towards this goal is *scalable cryptography*: In many cryptographic schemes currently deployed, the security degrades linearly with the number of instances (e.g. encrypted messages) in the system. As this number can be huge in contexts like cloud computing, the parameters of the scheme have to be chosen considerably larger—and in particular depending on the expected number of instances in the system—to maintain security guarantees. We advance the state-of-the-art regarding scalable cryptography by constructing schemes where the security guarantees are independent of the number of instances. This allows to choose smaller parameters, even when the expected number of instances is immense.

- We construct the first *scalable encryption scheme* with security against active adversaries which has both compact public keys and ciphertexts. In particular, we significantly reduce the size of the public key to only about 3% of the key-size of the previously most efficient scalable encryption scheme. (Gay, Hofheinz, and Kohl, CRYPTO, 2017)

- We present a *scalable structure-preserving signature scheme* which improves both in terms of public-key and signature size compared to the previously best construction to about 40% and 56% of the sizes, respectively. (Gay, Hofheinz, Kohl, and Pan, EUROCRYPT, 2018)

Another important area of cryptography is *secure multi-party computation*, where the goal is to jointly evaluate some function while keeping each party's input private.

In traditional approaches towards secure multi-party computation either the communication complexity scales linearly in the size of the function, or the computational efficiency is poor. To overcome this issue, Boyle, Gilboa, and Ishai (CRYPTO, 2016) introduced the notion of *homomorphic secret sharing*. Here, inputs are shared between parties such that each party does not learn anything about the input, and such that the parties can locally evaluate functions on the shares. Homomorphic secret sharing implies secure computation where the communication complexity only depends on the size of the inputs, which is typically much smaller than the size of the function.

A different approach towards efficient secure computation is to split the protocol into an input-independent *preprocessing phase*, where long correlated strings are generated, and a very efficient *online phase*. One example for a useful correlation are *authenticated Beaver triples*, which allow to perform efficient multiplications in the online phase such that privacy of the inputs is preserved and parties deviating the protocol can be detected. The currently most efficient protocols implementing the preprocessing phase require communication *linear* in the number of triples to be generated. This results typically in high communication costs, as the online phase requires at least one authenticated Beaver triple per multiplication.

We advance the state-of-the art regarding efficient protocols for secure computation with low communication complexity as follows.

- We construct the first *homomorphic secret sharing scheme* for computing arbitrary functions in NC^1 (that is, functions that are computably by circuits with logarithmic depth) which supports message spaces of arbitrary size, has only negligible correctness error, and does not require expensive multiplication on ciphertexts. (Boyle, Kohl, and Scholl, EUROCRYPT, 2019)

- We introduce the notion of a *pseudorandom correlation generator* for general correlations. Pseudorandom correlation generators allow to *locally* extend short correlated seeds into long pseudorandom correlated strings. We show that pseudorandom correlation generators can replace the preprocessing phase in many protocols, leading to a preprocessing phase with *sublinear* communication complexity. We show connections to homomorphic secret sharing schemes and give the first instantiation of pseudorandom correlation generators for authenticated Beaver triples at reasonable computational efficiency. (Boyle, Couteau, Gilboa, Ishai, Kohl, and Scholl, CRYPTO, 2019)

Chapter 1

Introduction

Due to the unparalleled rise of the Internet, cryptography has become essential in our everyday life. Often unnoticed, cryptography takes care of many tasks we take for granted: Cryptographic protocols ensure, for instance, that we have control over *who we talk to* and *who listens to our conversations*.

A cryptographic milestone was the discovery of public-key cryptography in the 1970s, manifested in the protocol for secure key exchange published by Diffie and Hellman in [DH76]. Prior to this discovery, all secure digital communication required exchanging a secret via a secure channel ahead of time. This was feasible when cryptography served mostly military purposes, but the world of today would be unimaginable without public-key cryptography:

- *Public-key encryption schemes* enables secure communication via a public channel, where messages can be eavesdropped or even altered *without* an a priori shared secret.
- *Signatures* ensure authentication and integrity, by making it *publicly verifiable* that a message was sent by a certain person and is unaltered.

A major challenge we are facing today is privacy in the age of internet surveillance. Without cryptography, we would be completely transparent to any eavesdropping entity. Encryption solves the problem for personal communication, but many problems require more advanced solutions. Consider for instance database queries: Looking up an entry in an analog encyclopedia cannot be easily observed by third parties, whereas from search queries to an online database one can derive possibly critical information.

On top of reestablishing the premises of the physical world, cryptography can solve tasks seemingly impossible. Cryptography allows researchers to compute on medical data—for example, to find connections between genetic preconditions and certain diseases—without learning anything apart from the considered correlation, thereby preserving the parties' privacy. This falls in the area of secure computation:

- *Secure multi-party computation* allows parties to evaluate a function on secret inputs, such that each party learns nothing about the inputs of the other parties except from what can be trivially derived from the function output.

For cryptographic research it is not sufficient to merely provide feasibility results; in order to be utilized in practice and therefore have an impact on the real world, cryptographic protocols are required to be efficient. The two main measures to consider are computational and communication complexity. Computational complexity describes the required resources, usually time and memory, used to execute the protocol, whereas communication complexity specifies the size and number of messages needed to be sent between the involved parties.

In many current cryptographic protocols, communication complexity is the bottleneck: Protocols are expected to work even over low-bandwidth connections with minimal latency for the parties involved; at the same time electronic devices come with increasing computational power.

During my PhD, I have participated in advancing the state-of-the-art regarding communication complexity in cryptographic protocols: on the one hand by constructing schemes for secure communication with shorter concrete parameters, on the other hand by developing protocols which allow to reduce the communication in protocols for secure computation of a broad class of functions considerably.

Improving Concrete Parameter Sizes. For long-term security guarantees, in particular for sensitive applications like electronic voting, one is generally interested in *provably* secure protocols instead of ad-hoc constructions. Unfortunately, unconditional security is impossible to achieve in many contexts—such as public-key cryptography—without solving long-standing open problems in complexity theory. For this reason, one usually settles for reducing the security of a cryptographic scheme to the hardness of a well-known problem, as, for instance, factoring the product of two large primes.

We consider a problem hard, if no adversary is able to solve it *efficiently*. As computational power increases over time, also the notion of efficiency changes. For this reason, we parametrize security by a *security parameter* λ. Currently, a problem is considered hard if any (known) adversary requires at least 2^{100} or 2^{128} steps to solve it. Note that this is referred to 100-bit or 128-bit security, respectively. To clarify: Even the best publicly known supercomputer would take currently more than a thousand centuries to break a problem satisfying 100-bit security.[1]

Security reductions from the security of a cryptographic scheme to the hardness of the underlying problem often come with a *loss* L, that is, an adversary breaking the scheme with some advantage ε, will break the underlying problem with success probability ε/L. This loss L does critically affect the parameter choice for using the scheme in practice.

For a typical example, consider a signature scheme that is secure based on the factoring assumption with a security loss that equals the number of signatures queried by the adversary. For choosing the parameters for a concrete instantiation, the security loss has to be taken into account: In typical applications the number of signatures can be in the order of 2^{30}, therefore, in order to achieve 128-bit security for the signature scheme, we have to choose the length of the primes such that the factoring assumptions withstands 158-bit attacks. In particular, in the described example the concrete size of the public key and signatures crucially depends on the number of expected signatures to be released. As this number might not even be known at time of set-up, either one has to find an upper bound (thereby unnecessarily increasing the size of the parameters) or possibly compromise long-term security.

A desirable goal is therefore to construct schemes with *tight security reductions*, where the loss is constant or at most linear in the security parameter and in particular independent of the number of instances in the system.[2]

This allows to choose parameters regardless of the number of signatures or encryp-

[1] As of November 2018, `https://www.top500.org/lists/2018/11/`.

[2] Note that to distinguish between a constant security loss and a loss linear in the security parameter, the latter is sometimes referred to as *almost tight reduction*.

tions to be issued, while at the same time preserving *provable* security guarantees. This leads to shorter parameters in practice, thereby reducing the communication complexity in protocols building on these schemes. Unfortunately, tightly secure schemes are usually more difficult to construct and, as a result, often come with larger parameters to begin with. Towards closing this gap, in this thesis the state-of-the art is improved as follows:

- We construct the first actively[3] secure encryption scheme with short public key *and* ciphertexts, whose security can be tightly reduced to the underlying assumption. Our scheme has a public-key size of only about 3% of the key size of the previously best tight construction. In particular, we significantly reduce the price to pay for scalable security by almost closing the gap to the most efficient non-tight construction: Our encryption scheme has an overhead of only *one* group element in the ciphertexts and *four* group elements in the public key compared to the scheme of Kurosawa and Desmedt [KD04]. [GHK17]

- We present a *structure-preserving*[4] *signature scheme* with tight security reduction, improving on the previous work in terms of public-key and signature sizes (to about 40% and 56% of the sizes, respectively). [GHKP18]

Exploring New Ways Towards Succinct Secure Computation. In the setting of secure computation, a change of the underlying paradigm of a cryptographic protocol can lead to much more drastic improvements in terms of communication complexity than can be achieved by merely reducing the size of parameters.

The most famous example for secure computation is the Millionaires' Problem, introduced by Yao in 1982 [Yao82a]: Two millionaires want to find out who is richer, without leaking anything apart from this one bit of information to the other party. Whereas the first solution proposed by Yao was not yet efficient, four years later he proposed a protocol that allows to securely evaluate general circuits via *garbling* [Yao86]: The idea is, roughly, that one party sends an encrypted circuit to the other party together with the keys necessary for evaluation, such that the other party learns the circuit output and nothing else. While this approach is computationally quite efficient, the communication complexity scales with the size of the circuit, which is typically large, and in particular much larger than the input sizes. This approach is thus not practical when restricted to low-bandwidth networks.

Much later, in [Gen09] Gentry constructed one of the long-searched for holy grails of cryptography, *fully homomorphic encryption*. Fully homomorphic encryption allows the evaluation of *arbitrary* circuits on encrypted messages, thereby yielding a protocol for secure computation with communication time linear in the input plus output size: In short, one party can send its encrypted input to the other party, who can homomorphically evaluate the circuit on the ciphertexts and send back the encrypted output to the first party, who can decrypt to obtain the output. Unfortunately, even the most current optimized procedures for multiplication on ciphertexts are computationally expensive, and therefore the practical applicability of this ap-

[3] Here, active refers to the ability of adversaries to *actively* alter messages, whereas pure eavesdropping attacks are usually referred to as *passive*.

[4] A signature scheme is called structure-preserving, if it obeys a certain structure which makes it nicely compose with other cryptographic building blocks.

proach limited. Further, constructions of fully homomorphic encryption are only known from a narrow set of so-called lattice-based assumptions.

To cope with these limitations, Boyle, Gilboa, and Ishai [BGI16a] took a different path to succinct secure computation via *homomorphic secret sharing (HSS)*. Homomorphic secret sharing can be viewed as a relaxation of fully homomorphic encryption, where the inputs are separated into so-called secret shares such that a single secret share completely hides the input, and such that further functions can be *locally* evaluated on shares.

A very simple example of homomorphic secret sharing for the class of *linear functions* is additive secret sharing: Here, a value $s \in \mathbb{Z}_r$ is shared into a and b such that $a + b = s \mod r$ for some natural number r. Additive secret sharing allows the parties to locally evaluate *linear* functions on the shares, i.e. functions f for which $f(a) + f(b) = f(a + b) \mod r$.

To allow evaluating more complex classes of functions, Boyle et al. [BGI16a] construct a homomorphic secret sharing for all functions in NC^1 (i.e. the class of function that can be computed by circuits with logarithmic depth) based on the decisional Diffie-Hellman assumption, thereby breaking the circuit-size barrier for secure computation based on a discrete logarithm type of assumption.

Unfortunately, their approach requires an expensive *share-conversion* operation to be performed after each multiplication. Additionally, the share-conversion introduces an inverse-polynomial[5] correctness error and restricts their construction to polynomial-sized input spaces.

On the other hand, even though homomorphic secret sharing was introduced as a relaxation of fully-homomorphic encryption, all prior constructions based on lattices build atop specific forms of fully-homomorphic encryption, and in particular require expensive multiplications on ciphertexts [AJL$^+$12, BGI15, DHRW16, BGI$^+$18].

- We construct the first *homomorphic secret sharing scheme* based on lattices that supports evaluation of arbitrary functions in NC^1, supports an arbitrary message space, comes with negligible[6] correctness error and does not require ciphertext multiplications. We show the practical applicability of our scheme for the use case of privacy-preserving counting queries on distributed databases. [BKS19]

In another line of work within secure computation, the protocol is split up in an input-independent preprocessing phase, and a fast online phase [Bea92, BDOZ11, DPSZ12, NNOB12, DZ13]. This approach is particularly appealing, when protocols are required to be fast once the inputs are known, but can tolerate a comparatively expensive preprocessing phase, as for instance electronic voting schemes.

The idea underlying these works is that the parties jointly set up long correlated strings in the preprocessing phase, which are then used to implement a fast online phase. One example are so-called "Beaver triples" [Bea92], which allow fast multiplication of shared values in the online phase. However, in existing works, the preprocessing phase is usually very communication intensive and requires the parties to save the long correlated strings until the online phase is initiated. A major

[5] in the security parameter

[6] A function is called negligible, if it vanishes faster than any inverse polynomial.

open question—leading to direct efficiency improvements in practice—is therefore, how to perform preprocessing with less communication.

- We introduce the primitive of *pseudorandom correlation generators* for general correlations. A pseudorandom correlation generator allows two parties to extend short correlated seeds into long correlated pseudorandom strings. We give connections to homomorphic secret sharing schemes and provide concrete instantiations for a number of useful correlations. We show that a pseudorandom correlation generator can replace the preprocessing phase in many protocols for secure computation, for instance in the protocol of Damgård et al. [DPSZ12].

  Pseudorandom correlation generators can lead to a major improvement regarding communication complexity, as jointly setting up short correlated seed requires significantly less communication than generating long correlated strings from scratch. Further, with our approach the parties only need to save a short seed, which they can *silently* (that is, without any further communication) expand before engaging in an online protocol. Our work is the first construction of practically efficient *silent* preprocessing for general purpose protocols for secure computation. [BCG+19b]

Other Results. In the following I want to elaborate on related topics I worked on during my time as a PhD student.

As public-key encryption is generally much less efficient than symmetric-key encryption where the secret key is used both for en- and decryption, in real-world applications usually a hybrid encryption scheme is employed: Namely, first a common key is established which can then be used for further communication. One primitive to do so is *non-interactive key exchange*, where the parties can derive a common key *merely* having knowledge of the other party's public key. In general, non-interactive primitives are desirable as they do not require an extra round of communication, but therefore typically more challenging to construct: Indeed, Bader et al. [BJLS16] showed that known non-interactive schemes like the Diffie-Hellman key exchange [DH76] have a security loss *inherently quadratic* in the number of users. As the number of users in real-world applications is typically large (say 2^{30}), this loss critically affects the group size and therefore the concrete efficiency of the scheme.

- We give the first construction of *non-interactive key-exchange* with security loss *linear* in the number of users. Further, we improve the bound of [BJLS16] and show that for our scheme (and related schemes) this loss is inherent. This gives a strong indication that constructing tightly secure non-interactive key exchange is inherently difficult and seems therefore not the most promising approach towards constructing schemes with better concrete efficiency. [HHK18]

Another fundamental primitive is the notion of a *pseudorandom function*, which is useful, for instance, to construct symmetric-key encryption. As the name suggests, a pseudorandom function is a keyed function whose output looks random to any bounded adversary given black-box access. Beyond secure communication, pseudorandom functions have numerous applications. Extending pseudorandom functions, Micali, Rabin, and Vadhan [MRV99] introduced the notion of a *verifiable random function*: Here, the key holder can additionally prove correct function evaluation. This seeming contradiction to the pseudorandomness requirement can be

achieved as follows. The key holder commits to the secret key (e.g. by giving it in the exponent), and at time of evaluation provides a proof of correct evaluation respective this commitment. The security requirement is quite strong: Even a maliciously set-up key is required to commit the key holder to a *unique* function. While this primitive finds many applications, for instance in the area of electronic cash [MR02, ASM07, BCKL09], the requirement of unique provability makes it difficult to construct. Up to the work of Hofheinz and Jager [HJ16] in 2016, no construction with security reduction to a *standard* assumption (that is, hardness assumptions of problems that are well-studied and believed to be difficult to solve) was known that supports an exponential-size input space and achieves full adaptive[7] security. On the downside, their construction comes with a proof size *linear* in the input length.

- We give the first construction of a *verifiable random function* with security based on a *standard assumption*, exponential-size input space and full adaptive security, where proofs only comprise a *sublinear* number of group elements. Our proof length is *independent of the input length.* Moreover, our construction comes with the shortest proofs to date, even compared to constructions based on non-standard assumptions, at the price of a larger public-key size.[8] [Koh19]

Regarding secure computation, one of the most basic primitives is *oblivious transfer (OT)* [Rab81, EGL85]. Oblivious transfer allows one party to receive 1-out-of-2 messages from another party such that the first party learns nothing about the other message, and the second party learns nothing about the choice of the first party. It was shown by Kilian [Kil88] that oblivious transfer implies secure computation of arbitrary functions. Moreover, many protocols for efficient secure computation today are built upon oblivious transfer. As these protocols typically require many OTs (say millions), the construction of efficient oblivious transfer is an important task for cryptographers. Similar to encryption, the most efficient protocols to-date follow a hybrid approach called *OT-extension*: Assuming black-box access to a number of base-OTs, many OTs are constructed using only cheap symmetric-key cryptography. This approach was introduced by Beaver [Bea96], who constructed a 2-round *OT-extension* protocol, but yet with poor concrete efficiency. Later, Ishai et al. [IKNP03] presented the first efficient (but 3-round) protocol, whose paradigm, in fact, still underlies the most efficient constructions of OT-extension today.

- Building upon the pseudorandom correlation generators of our work [BCG+19b], we construct the first *efficient* 2*-round OT-extension protocol.* For the use case of generating random OTs (that is, OTs with random choice bit and messages) our approach improves communication complexity by more than a factor of thousand compared to the protocol by Ishai et al. [IKNP03] for 10 million OTs. Our protocol comes at the price of reduced computational efficiency, but we still perform favorably over wide area network connections. In fact, over networks with a speed of 10 MBps, we improve the state-of-the-art time cost by almost a factor of 50 for setting up 10 million random OTs. Further, we give

[7] Full adaptive adversaries are given access to the public key and can adaptively choose function queries depending on previous output values, thereby best modeling reality.

[8] Note that for comparison we consider only constructions that support exponential-size input space and come with full adaptive security.

a protocol achieving security against active adversaries at low additional cost (estimated 10 – 20%). [BCG+19a]

Technical Overview

In the following we give a high-level overview of the techniques involved in this thesis.

Tight Security Reductions

The first to point out the importance of tight security reductions were Bellare, Boldyreva, and Micali [BBM00]. Nevertheless, many state-of-the-art constructions come with non-tight security reductions, as constructing schemes that allow tight security reductions is challenging.

Why Proving Tight Security is Difficult. Broadly speaking, a security reduction transforms an adversary attacking the cryptographic scheme into an adversary solving the underlying problem. To understand the difficulty of tight security reductions, consider the example of signature schemes: We formulate security as an experiment, where the adversary gets the public key and can adaptively query signatures for an arbitrary number of chosen messages and finally has to provide a forgery. We say a signature scheme is secure, if no probabilistic polynomial-time adversary is able to successfully forge a signature for a fresh message with more than negligible success probability.

In order to transform such an adversary to an adversary solving some computational problem, the reduction has to answer all signature queries while at the same time being able to transform the signature given by a successful adversary into a solution to the underlying problem. In particular, the reduction cannot compute this signature itself, as otherwise it would efficiently solve the underlying problem, contradicting its hardness. This difficulty is captured by the work of Coron [Cor02], who proves that the security reduction of any signature scheme with *unique signatures* to a non-interactive assumption has an inherent security loss in the number of signing queries asked by the adversary.

A way to circumvent this lower bound is to allow every message to have several valid signatures and partition the space of signatures such that

- for each message the reduction can produce a valid signature and
- for each message the reduction can turn the signature provided by the adversary with constant probability into a solution to the underlying problem.

In particular, signatures the security reduction can generate and signatures that can be transformed to a solution have to be indistinguishable to the adversary.

Constructing CCA-Secure Encryption Schemes. Going back to encryption, constructing public-key encryption schemes which satisfy security against active adversaries or more formally security against *chosen-ciphertext attacks (CCA)*, is challenging even ignoring tightness. The ability to alter messages is modeled by giving the adversary access to a decryption oracle, to which it can query arbitrary ciphertexts. Finally, the adversary has to decide to which of his chosen messages was encrypted by the security experiment (of course without using the decryption oracle on the challenge ciphertext).

Implementing the decryption oracle imposes a problem on the security reduction: A reduction that knows the secret key cannot make use of the underlying adversary (as it can decrypt the challenge message itself), while not in knowledge of the secret key it cannot answer decryption queries of the adversary. For this reason the first public-key encryption scheme with security against active adversaries was given more than 10 years after the birth of public-key cryptography by Naor and Yung [NY90]. The idea is to make the challenge ciphertext *inconsistent*, while allowing only *consistent* queries to the decryption oracle, by adding a proof of well-formedness to the ciphertexts.

Later, Cramer and Shoup [CS98] gave the first efficient construction, building on a primitive they called *hash proof system*. A hash proof system for an NP language $\mathcal{L} \subseteq \mathcal{X}$ provides two different ways to generate a key k given a value $x \in \mathcal{L}$: Either given the public key and a witness for $x \in \mathcal{L}$, or given the secret key only. Further, for *correctness* it is required that both methods yield the same key, and for *security* (also referred to as 1-*universality*) that even given access to the public key, the secret evaluation for any $x \in \mathcal{X} \setminus \mathcal{L}$ is distributed uniformly at random over the key space. In particular, this means that even an *unbounded* adversary cannot extract the secret key from the public key. In other words, even with knowledge of the public key the secret key has some entropy (i.e. unpredictability) left.

Building upon the techniques of [CS98], in 2004 Kurosawa and Desmedt [KD04] presented an encryption scheme, which is the most efficient CCA-secure encryption scheme in this line of work until today: They replace the hash proof system of [CS98] by a 2-*universal* hash proof system, for which security is guaranteed even seeing the secret evaluation of one $x \in \mathcal{X} \setminus \mathcal{L}$. Their public key consists of the hash proof system public key, and ciphertexts are of the form $(x, \mathsf{Enc}_k(M))$, where $x \in \mathcal{L}$, k is the hash proof key corresponding to $x \in \mathcal{L}$ and Enc is a symmetric authenticated encryption scheme.

In the proof of security the key in the challenge ciphertext is switched to random (by switching $x \in \mathcal{L}$ to $x \in \mathcal{X} \setminus \mathcal{L}$), while decryption queries with $x \notin \mathcal{L}$ can be rejected: The only information the adversary has about the secret key is via the public key and the single challenge query, thus an adversary submitting a *valid* decryption query outside $\mathcal{L}$ contradicts 2-universality. Therefore, via the entropy left in the hash proof system secret key (which is neither leaked by the public key nor by decryption queries in $\mathcal{L}$), the key in the challenge ciphertext looks random from the adversaries point of view and therefore perfectly hides which of the challenge messages is encrypted.

Towards Tightly Secure Encryption. Going from single- to multi-ciphertext security with this approach though, where the adversary is allowed to ask many challenge queries, a problem occurs. Classically, single-ciphertext security implies multi-ciphertext security via a hybrid argument, but this entails a security loss proportional to the number of challenge ciphertexts. The problem proving a tight security reduction is due to the following: The entropy necessary to reject decryption queries outside $\mathcal{L}$ is limited, namely the entropy left in the hash proof system secret key given the corresponding public key and one statement outside $\mathcal{L}$. In particular, this entropy is not sufficient when many challenge ciphertexts are switched to random. For the proof to carry over to the multi-ciphertext setting, one would require $(Q+1)$-universality of the underlying hash proof system, where Q is the number of

challenge ciphertexts asked by the adversary, which is a priori unbounded.

To overcome this issue, Gay et al. [GHKW16] replace the 2-universal hash proof system by a λ-universal hash proof system, which can be thought of a combination of λ individual 1-universal hash proof systems (where the i-th hash proof system secret key is chosen depending on the i-th bit of the statement x in bit representation[9]). Now, the secret key can be gradually randomized via a partitioning argument. Roughly, in the i-th step the security reduction proceeds as follow: Half the ciphertexts are switched to some language $\mathcal{L}_0$, and the other half to some language $\mathcal{L}_1$ (depending on the i-th bit of the statement x itself—note that the circularity issue arising can be resolved via re-sampling or by only hashing part of the statement). Then, the hash proof system key is randomized by adding a random offset that shows up in all ciphertexts (i.e. challenge ciphertexts and decryption queries) with $x \in \mathcal{L}_0$, and, subsequently, another random offset that shows up in all ciphertexts with $x \in \mathcal{L}_1$. The λ-universality of the hash proof system ensures that during randomization *ill-formed* decryption queries (that is, roughly, decryption queries that do not follow the partitioning) can be rejected. After λ steps, each ciphertext has a *freshly* randomized secret key, and thus all decryption queries outside $\mathcal{L}$ can be rejected, again via an entropy argument. Finally, all challenge ciphertexts can be switched to random at once, using the re-randomizability of the decisional Diffie-Hellman assumption.

The problem of this approach is that the partitioning is *static* depending on the i-th bit of the statement itself with all entropy for partitioning part of the public key. This entails a large public key (of size *linear* in the security parameter).

How to Achieve Compact Parameters. In our work [GHK17] we follow the basic idea of Gay et al. [GHKW16], but use the *adaptive partitioning* strategy of Hofheinz [Hof12] to achieve short ciphertexts *and* public keys. In [Hof12] the partitioning dynamically depends on a single bit of information in the ciphertext, which can be used for different partitionings throughout the proof. In particular, proving well-formedness of ciphertexts now only requires a proof about this bit of information, leading to significantly shorter keys. On the other hand, the statement to be proven about this bit is *non-linear*, and thus requires a more complex proof system. For this reason, the work of Hofheinz [Hof12] uses an explicit pairing[10]-based designated verifier proof of an or-like language (i.e. the disjunction of two languages). This has two shortcomings: First, evaluating pairings is expensive and second, the proof itself adds an extra 3 elements to the ciphertexts.

A main challenge of our work was thus to construct a compact pairing-free proof for an or-like language. In the following we give an overview of the proof for the disjunction $\mathcal{L} \cup \mathcal{L}_0$. Observe that honestly generated ciphertexts will *always* lie in $\mathcal{L}$ (switching to $x \in \mathcal{L}_0$ is only necessary during the security reduction). It is thus sufficient to construct a proof system, such that only statements $x \in \mathcal{L}$ can be proven honestly. The difficulty is to ensure that even an adversary given access to an arbitrary number of proofs for $x \in \mathcal{L}_0$ cannot successfully forge a proof for $x \notin \mathcal{L} \cup \mathcal{L}_0$. We solve this by combining two hash proof system evaluations in such a way, that one "blinds" the other *if and only if* $x \in \mathcal{L}_0$. Thus, by randomizing one

[9] To obtain a λ-bit representation of x, one can apply a suitable collision resistant hash function to x.

[10] A pairing is a non-degenerate bilinear mapping from two groups into a target group.

hash proof system evaluation, the entropy in the other hash proof system secret key is protected even given *many* proofs for statements $x \in \mathcal{L}_0$.

The resulting proof system adds only *one* extra group element to ciphertexts. We therefore almost meet the original scheme of Kurosawa and Desmedt in terms of ciphertext *and* public-key size.

Structure-Preserving Signatures. Next, we turn our attention to a special type of signatures: A signature scheme is called *structure preserving* if its public keys, messages and signatures consist only of group elements, and verification can be expressed as equations over some cyclic group, which is usually required to be pairing-friendly. The motivation being that structure-preserving signatures nicely combine with the efficient non-interactive zero knowledge proofs by Groth and Sahai [GS08] and similar proof systems. This makes them widely applicable, for instance as key ingredient in building efficient group signatures [LPY15], anonymous credential systems [BCKL08] and electronic voting schemes [GL07].

Despite their importance, before our work there existed only two constructions of structure-preserving signature schemes with *tight* security reduction to the underlying problem [HJ12, AHN+17]. Moreover, both come with significantly worse parameters than non-tight structure-preserving signatures or even tight standard signature schemes. The difficulty in constructing structure-preserving signatures with tight security reductions seems to stem from the partitioning techniques usually applied, which do not easily carry over to the structure-preserving setting.

In [GHKP18] we show how to obtain structure-preserving signatures with significantly shorter parameters, building on techniques described previously. Further, we improve the security loss from $\mathcal{O}(\lambda)$ to $\mathcal{O}(\log Q)$, where λ is the security parameter, and Q is the number of signing queries asked by the adversary (polynomial in λ).

From Encryption to Signatures. In order to make our techniques from [GHK17] applicable to a broader setting, we distill the main randomization technique in a *core lemma*: Given a hash proof system for a language $\mathcal{L}_0$, together with an or-proof for the language $\mathcal{L}_0 \cup \mathcal{L}_1$, we can gradually randomize the hash proof system secret key, such that in the end each instance (later corresponding to a signature) has an individually randomized secret key. We observe that by enumerating the number of instances from 1 to Q in the reduction, we can improve the tightness from λ to $\log Q$ by partitioning according to the counter instead of applying a collision resistant hash function to the instance.

The core lemma allows us to derive a *message authentication scheme (MAC)*—the symmetric analogue to a signature scheme, where the secret key is used for signing and verification. More precisely, our MAC is the linear combination of two hash proof system evaluations with the *message* as coefficient, where for our encryption scheme a collision resistant hash of the statement itself was used. The technique of transforming an encryption scheme (or, more precisely, a key encapsulation mechanism) into a MAC is due to Dodis et al. [DKPW12].

While this is an important step towards structure-preserving signatures, two problems are yet to address: First, verification is only possible to a *designated* verifier. And second, the described technique only supports authenticating messages in $\mathbb{Z}_p$ (where p is the group order) and is therefore not structure-preserving.

For public verification we build on the generic transformation of Bellare and Goldwasser [BG90]. Basically, the idea is to commit to the hash proof system secret key

Reference	$\lvert\mathsf{pk}\rvert$	$\lvert\mathsf{ct}\rvert - \lvert m\rvert$	sec. loss	assumption	pairing
[CS03]	3	3	$\mathcal{O}(Q)$	DDH	no
[KD04, HK07]	$k+1$	$k+1$	$\mathcal{O}(Q)$	k-LIN ($k \geq 1$)	no
[HJ12]	$O(1)$	$O(\lambda)$	$\mathcal{O}(1)$	DLIN	yes
[LJYP14, LPJY15]	$\mathcal{O}(\lambda)$	47	$\mathcal{O}(\lambda)$	DLIN	yes
[AHY15]	$\mathcal{O}(\lambda)$	12	$\mathcal{O}(\lambda)$	DLIN	yes
[GCD+16, HKS15]	$\mathcal{O}(\lambda)$	$6k$	$\mathcal{O}(\lambda)$	k-LIN ($k \geq 1$)	yes
[GHKW16]	$2\lambda k$	$3k$	$\mathcal{O}(\lambda)$	k-LIN ($k \geq 1$)	no
[Hof17]	$2k(k+5)$	$k+4$	$\mathcal{O}(\lambda)$	k-LIN ($k \geq 2$)	yes
[Hof17]	20	28	$\mathcal{O}(\lambda)$	DCR	—
Ours [GHK17]	6	3	$\mathcal{O}(\lambda)$	DDH	no
	$k^2(k+1)+4k$	$k(k+2)$	$\mathcal{O}(\lambda)$	k-LIN($k \geq 2$)	no

Figure 1.1: Comparison amongst CCA-secure encryption schemes, where Q is the number of ciphertexts, $\lvert\mathsf{pk}\rvert$ denotes the size (in groups elements) of the public key, and $\lvert\mathsf{ct}\rvert - \lvert m\rvert$ denotes the ciphertext overhead, ignoring smaller contributions from symmetric-key encryption. Here, DDH stands for the decisional Diffie-Hellman assumption, and k-LIN for the k-Linear assumption, corresponding to DDH for $k = 1$ and Decision Linear (DLIN) for $k = 2$.

in the public key, thereby making the hash proof system evaluation publicly verifiable via a pairing equation. Further, the or-proof used to achieve a tight security reduction in the core lemma has to be publicly verifiable. For this purpose we use an instantiation of Ràfols [Ràf15], building upon the techniques of Groth, Sahai, and Ostrovsky [GOS12]. Note that the publicly verifiable or-proof makes up the major part in our signatures, namely 10 out of 14 group elements, while for our encryption scheme one extra group element in the ciphertexts is sufficient.

Finally, in order to obtain signature structure-preserving signatures, we have to replace the 2-universal hash proof system by one that allows to embed messages in the underlying group. The idea is to replace the pairwise independent function underlying the hash proof system by a structure-preserving one, such that the core lemma is still applicable. This results in a structure-preserving signature scheme, whose security can be tightly reduced to the symmetric external Diffie-Hellman (SXDH) assumption. Note that the same technique was used already in the context of structure-preserving signatures by Kiltz, Pan, and Wee [KPW15], without yielding a tight security reduction though.

Related Work. It took more than 10 years from the introduction of tightness by Bellare, Boldyreva, and Micali [BBM00], until the first tightly secure encryption scheme was proposed by Hofheinz and Jager [HJ12]. More efficient schemes followed [ADK+13, CW13, BKP14, LJYP14, HKS15, LPJY15, AHY15, GCD+16, Hof16, Hof17, GHKW16]. The first pairing-free construction was proposed by Gay et al. [GHKW16], but with a public key consisting of about 200 group elements. We give an comparison to selected previous works in Figure 1.1 (taken almost verbatim from our work [GHK17]).

In case of structure-preserving signatures the gap regarding the instance size was even larger. The only efficient instantiation with tight security reduction before our work was due to Abe et al. [AHN+17] with 25 elements in the signature. For a comparison with selected tight and non-tight constructions we refer to Figure 1.2 (taken almost verbatim from our work [GHKP18]). Note that the construction of

Reference	$\|M\|$	$\|\sigma\|$	$\|\mathsf{pk}\|$	sec. loss	assumption
[HJ12]	1	$10\ell + 6$	13	8	DLIN
[ACD+16]	$(n_1, 0)$	$(7, 4)$	$(5, n_1 + 12)$	Q	SXDH, XDLIN
[LPY15]	$(n_1, 0)$	$(10, 1)$	$(16, 2n_1 + 5)$	$\mathcal{O}(Q)$	SXDH, XDLIN_2
[KPW15]	$(n_1, 0)$	$(6, 1)$	$(0, n_1 + 6)$	$2Q^2$	SXDH
[JR17]	$(n_1, 0)$	$(5, 1)$	$(0, n_1 + 6)$	$Q \log Q$	SXDH
[AHN+17]	$(n_1, 0)$	$(13, 12)$	$(18, n_1 + 11)$	80λ	SXDH
[JOR18]	$(n_1, 0)$	$(11, 6)$	$(7, n_1 + 16)$	116λ	SXDH
Ours [GHKP18]	$(n_1, 0)$	$(8, 6)$	$(2, n_1 + 9)$	$6 \log Q$	SXDH
[AJOR18]	$(n_1, 0)$	$(6, 6)$	$(n_1 + 11, 2n_1 + 12)$	$36 \log Q$	SXDH
[AJO+19]	$(n_1, 0)$	$(7, 4)$	$(2, n_1 + 11)$	$6 \log Q$	SXDH

Figure 1.2: Comparison of standard-model structure-preserving signature schemes (in their most efficient variants). Here, we restrict ourselves to unilateral schemes (with messages over $\mathbb{G}_1$). The notation (x_1, x_2) denotes x_1 elements in $\mathbb{G}_1$ and x_2 elements in $\mathbb{G}_2$. $|M|$, $|\sigma|$, and $|\mathsf{pk}|$ denote the size of messages, signatures, and public keys (measured in group elements). "Sec. loss" denotes the multiplicative factor that the security reduction to "assumption" loses, where we omit dominated and additive factors. (Here, "generic" means that only a proof in the generic group model is known.) For the tree-based scheme HJ12, ℓ denotes the depth of the tree (which limits the number of signing queries to 2^ℓ). Q denotes the number of adversarial signing queries, and λ is the security parameter.

Jutla, Ohkubo, and Roy [JOR18], which can be viewed as a direct improvement of [AHN+17], was concurrent and independent to our work.

Recent Developments. Quite recently, a number of works constructed tight public-key encryption satisfying additional security properties.

Building on our work [GHK17], Lyu et al. [LLHG18] give the first public-key encryption scheme with compact public key which enjoys tight simulation-based security under chosen-ciphertext and *selective-opening attacks*.[11]

Further, Han et al. [HLLG19] present a tight construction of a *leakage-resilient* public-key encryption, where the adversary is allowed to receive some leakage on the ciphertexts. This requires different techniques, as our proof system does not guarantee security when partial information of the secret key is leaked.

Abe et al. [AJOR18] provide the construction of a special kind of non-interactive proof system for linear-subspace languages with compact parameters that reduces tightly to the underlying assumption. This has applications both in public-key encryption and structure-preserving signatures: First, it gives a tight public-key encryption scheme with *public-verifiability* where everyone can verify that a given ciphertext decrypts to a valid plaintext. Second, it implies structure-preserving signatures with shorter signatures, but at the cost of larger public keys. Quite recently Abe et al. [AJO+19] constructed structure-preserving signatures comprising only 11 group elements and almost matching our public-key size.

Open Problems. There is still a gap between non-tight an tight constructions for both public-key encryption and structure-preserving signatures. For structure-preserving signatures it seems feasible to reduce the signature size further by constructing a more efficient publicly verifiable or-proof. It is not clear though, whether there is an inherent barrier to meet the size of non-tight constructions.

[11] A scheme is called selective-opening secure, if, given a set of ciphertexts and openings (including randomness) for a subset of these ciphertexts, the unopened ciphertexts are still hiding.

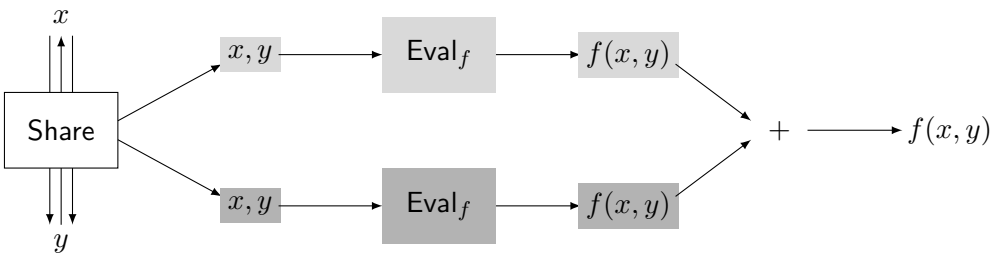

Figure 1.3: How to use homomorphic secret sharing for succinct secure computation. Here x and y denote the input values, and and the shares hold by parties P_0 and P_1, respectively. "Share" denotes a secure protocol for sharing the inputs and Eval_f denotes the procedure for locally evaluating the function f on the shares.

For encryption it is a very interesting open question whether there exist a tight construction which meets the ciphertext size of the scheme of Kurosawa and Desmedt (particularly with compact public key). This would either require a hash proof system like building block for the or-proof (with short public key)—with possibly interesting applications in other areas of cryptography, or require fundamentally new techniques allowing a tight security reduction.

Succinct Secure Computation

Regarding communication complexity of protocols for secure computation, there are two aspects to consider: The *size* of messages exchanged, as well as the *round complexity*, that is, the *number* of sequential messages sent between parties. In the following we give an overview of our advances constructing protocols with *succinct* communication, where in particular the size of the messages to be exchanged is independent of the circuit size of the function to be computed.

Homomorphic Secret Sharing. *Secret sharing* describes splitting up a secret into two (or more) shares, such that each share on its own does not leak anything about the secret, but altogether the secret can be recovered. *Homomorphic secret sharing* is a variant of secret sharing, where functions can be evaluated locally on the shares, resulting in a secret sharing of the function output. We consider the special case of homomorphic secret sharing with *additive reconstruction*, where the final secret sharing is required to be additive.

Following Boyle et al. [BGI16a], homomorphic secret sharing is one promising approach towards succinct secure computation: First, the parties engage in a protocol for securely sharing their input values. Note that the communication required for setting up the secret shares only depends on the input size and is in particular independent of the size of the circuit for computing the function. Now, the parties can locally evaluate the function on their respective shares and add up the output values to the final result. We give a depiction in Figure 1.3.[12]

Even though homomorphic secret sharing can be viewed as a relaxation of fully homomorphic encryption, somewhat surprisingly all previous constructions of homo-

[12]Figures taken (with minor modifications) from presentation by the author on "HSS from lattices without FHE", NYC Crypto Day, January 2019.

morphic secret sharing schemes based on lattices are actually at least as complex as fully-homomorphic encryption schemes.

In our work [BKS19] we present new techniques that yield the first lattice-based 2-party homomorphic secret sharing scheme for the class of functions in NC^1 that avoids expensive ciphertext multiplications. In the following we give an overview of the techniques involved.

Cryptography from Lattices. A *lattice* is a discrete subgroup of the n-dimensional space of real numbers. Lattices are interesting from a cryptographic view point, as certain problems in lattices (such as finding the shortest vector) are not known to be efficiently solvable. Lattice-based cryptography was kicked-off with the work of Ajtai [Ajt96], who gave the construction of collision resistant hash function which can be reduced to solving a lattice problem *in the worst case*. Since then numerous cryptographic building blocks have been constructed from lattices. Building cryptography on the hardness of lattice problems is particularly appealing, as—other than more classical assumptions like the decisional Diffie-Hellman assumption—lattice problems are not known to become significantly easier with quantum computers. Another reason for their popularity is that fully homomorphic encryption can be constructed from lattices [Gen09].

The idea underlying lattice-based encryption schemes is, roughly, that it is conjectured to be hard to distinguish a lattice vector which is perturbed (by adding a small noise) from a truly random vector. Using such a perturbed lattice vector to blind the message allows the holder of the secret key to recover the plaintext *up to the added noise*. In order to allow exact recovery of the message, one can, for instance, "blow up" messages multiplying a factor q/r, where q is the ciphertext modulus and r the plaintext modulus. It is relatively straightforward to see that (by the property of the underlying lattice) *adding* ciphertexts directly translates to adding the underlying plaintexts. For multiplication, on the other hand, one has to work a bit harder. In fact, to handle an unbounded number of multiplications, advanced and computationally expensive techniques such as *key-switching* and *modulus-reduction* are necessary. The problem is, roughly, that multiplication leads to a very rapid growth of the noise.

Homomorphic Secret Sharing without FHE. To avoid this source of inefficiency, an obvious question is: *Can we construct homomorphic secret sharing from lattices without fully homomorphic encryption?* Here, without fully homomorphic encryption refers to *avoiding* expensive ciphertext multiplication with the goal of *efficient* secure 2-party computation and 2-server private information retrieval.

In our work [BKS19], we answer this question affirmatively. Towards our construction, let us first take a closer look at homomorphic secret sharing from threshold fully homomorphic encryption. In threshold fully homomorphic encryption, key generation returns *secret key shares* $\mathsf{sk}_0, \mathsf{sk}_1$ together with a public key pk, such that $\mathsf{Dec}_{\mathsf{sk}_0}(\mathsf{Enc}_{\mathsf{pk}}(m)) + \mathsf{Dec}_{\mathsf{sk}_1}(\mathsf{Enc}_{\mathsf{pk}}(m)) = m \mod r$. Now, giving each party one of the secret key shares allows to homomorphically evaluate a function on the shares, and finally recovering the output by exchanging the final shares. Note that even though both parties hold a secret key share *throughout evaluation*, it is *only* used for decryption in the end.

We observe that we can avoid expensive ciphertext multiplication by using the secret key share also *during* evaluation itself. More precisely, we replace multiplica-

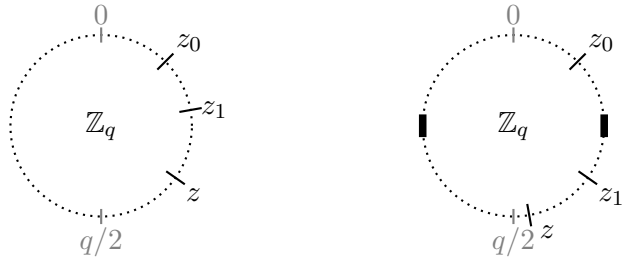

Figure 1.4: Graphic representation of *rounding*. Here, $z_0 \leftarrow \mathbb{Z}_q$ and $z_1 = z - z_0 \mod q$. On the left, $z \in \mathbb{Z}_q$ is arbitrary (which, in general, leads to a rounding error with *constant* probability). On the right, $z \approx (q/2) \cdot x \mod q$ for $x = 1$. In this case, rounding is successful unless z_0 falls into the *bad* area marked in black.

tion by a distributed decryption. This has several advantages: First, we do not have to deal with noise growth. Depending on the setting this allows to choose smaller parameters for the underlying ring, and therefore leads to a reduction in communication complexity compared to approaches from fully (or somewhat) homomorphic encryption. Second, distributed decryption turns out to be more efficient in practice, which on its own can lead to an estimated speed-up of about an order of magnitude for carrying out multiplications.

To understand our approach, note that decryption in lattice-based schemes is usually of the following form (e.g. [LPR13]): First, a bilinear function (say LDec) is applied to the secret key and the ciphertext. This yields $(q/r) \cdot x + e \mod q$, where x is the plaintext and e is a small noise term. Finally, rounding to the closest multiple of q/r allows to recover the plaintext. Let $\mathsf{sk}_0, \mathsf{sk}_1$ be key shares of the secret key with $\mathsf{sk}_0 + \mathsf{sk}_1 = \mathsf{sk} \mod q$ and c an encryption of x. Then, by bilinearity of LDec, we have

$$\mathsf{LDec}_{\mathsf{sk}_0}(c) + \mathsf{LDec}_{\mathsf{sk}_1}(c) = (q/r) \cdot x + e \mod q,$$

and, similarly, for "small" y it holds

$$\mathsf{LDec}_{y \cdot \mathsf{sk}_0}(c) + \mathsf{LDec}_{y \cdot \mathsf{sk}_1}(c) = (q/r) \cdot x \cdot y + e \cdot y \approx (q/r) \cdot x \cdot y \mod q.$$

This is the basic idea, of how giving the parties key shares of $y \cdot \mathsf{sk}$, allows to perform a multiplication via *distributed decryption*.

In order to recover the exact message though, it is left to perform the rounding on the shares. In general, given a sum $z_0 + z_1 = z \mod q$, rounding on secret shares yields an error with *constant* probability. Luckily, in our case we know $z \approx (q/r) \cdot x$ for some x which allows to apply a trick of Dodis et al. [DHRW16]: If $r \ll q$, $z_0 \leftarrow_R \mathbb{Z}_q$, $z_1 = z - z_0 \mod q$ and $z \approx (q/r) \cdot x$, then with high probability over the choice of z_0 it holds $\mathsf{Round}(z_0) + \mathsf{Round}(z_1) = \mathsf{Round}(z) \mod q$. For a graphic representation (for $r = 2$), we refer to Figure 1.4.[12]

This is still not sufficient, as it only allows to perform *one* multiplication on ciphertexts. In order to perform another multiplication, the parties need shares of $x \cdot y \cdot \mathsf{sk} \mod q$ (instead of $x \cdot y \mod r$). Introducing sk into the equation is comparatively simple by encrypting $x \cdot \mathsf{sk}$ instead of x alone. In general, encrypting the secret key itself requires a *stronger assumption*, namely that security of the encryption scheme holds even for messages depending on the secret key. In our

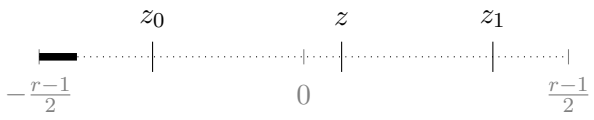

Figure 1.5: Graphic representation of *lifting*. Here, $z_0 \leftarrow \mathbb{Z}_r$, $z_1 = z - z_0 \mod r$ for $|z| \ll r$. Lifting is successful unless z_0 falls into the *bad* area marked in black.

context, though, this comes for free: As observed by Brakerski et al. in [BV11] encryption schemes with *nearly linear decryption* (as described) already satisfy this stronger security notion. Even more, we observe that generating encryptions of $x \cdot \mathsf{sk}$ is straightforward even without knowledge of the secret-key, allowing parties to share their inputs e.g. in the setting of secure computation.

A problem more challenging is the change of modulus. As the secret key shares are modulo r, we would have to continue with a ciphertext *modulo* r and plaintext *modulo* r_1 for some $r_1 \ll r$ for the next multiplication. This would lead to a leveled HSS scheme $q \gg r \gg r_1 \gg \cdots \gg r_\ell$ with *superpolynomial* drop in each level (in order for the rounding to work)—in particular, we would only be able to perform a *constant* number of multiplications somewhat efficiently.

We avoid the above conundrum, by (quite literally) doing *nothing*.[13] More precisely, we observe that whenever the plaintext is much smaller than the plaintext modulus (more precisely, when $|x \cdot y \cdot \mathsf{sk}| \ll r$), we can switch the modulus to any modulus of our choice (here q), again with negligible correctness error. We call this technique *lifting*. The idea is that if $|x \cdot y \cdot \mathsf{sk}|$ is small, then a random sharing of $x \cdot y \cdot \mathsf{sk} \mod r$ is with high probability actually a sharing over $\mathbb{Z}$ and thus correctness is preserved respective arbitrary moduli. For a graphic representation we refer to Figure 1.5.[12]

Altogether, our HSS has three levels: $B \ll r \ll q/B$. As long as the magnitude of our program (that is, all intermediary evaluation steps) are bounded by B, we are guaranteed correctness *with overwhelming probability*. Note that we only support *restricted multiplications*, where one factor is an input value. This limits our constructions to functions that can be computed by so-called *restricted multiplication straight-line programs*. This includes for instance all functions that can by computed by circuits of logarithmic depth, and branching programs.

Application of our HSS Scheme to Private Information Retrieval. For an efficiency comparison to somewhat homomorphic encryption (where only a bounded number of multiplications are supported) we consider generalized private-information retrieval. *Private information retrieval (PIR)* was introduced by Chor et al. in [CGKS95] and is one approach towards privacy-preserving database queries: Assuming servers that do not communicate with each other each holding a copy of the database, PIR allows to query the database without an individual server receiving any information about the content of the queries. Here, we consider the special case of 2-server PIR. While simple queries (like equality conditions and range-queries) can be efficiently supported by HSS construction for the family of point functions [GI14, BGI16b, WYG+17], our HSS allows to support more advanced queries, as for instance counting queries, conjunctive keyword search and pattern matching. As

[13] Sentence taken verbatim from [BKS19].

Depth	N	$\log q$	security
1	4096	102	145.1
5	4096	164	85.53
10	8192	252	111.9

B	N	$\log q$	security
2	4096	137	103.3
2^{64}	8192	267	104.9
2^{256}	16384	655	84.60

Figure 1.6: Private information retrieval for conjunctive keyword counting query, comparison. On the left hand side parameter choice for BFV ([Bra12, FV12]) with plaintext modulus 2. (Here, 10 gives a lower bound on the required depth.) "N" denotes the dimension of the underlying ring, "q" the ciphertext modulus and "security" the security level (in bit). On the right hand side we give the parameter choice for our HSS, where B denotes the program magnitude. As we do not have to account for the noise growth we get by with smaller parameters.

example consider the following. Given a database with entries of length 128 bit, each associated with a list of 10 keywords and given a list of 4 keywords: How long does it take to privately count all database entries which are associated to all 4 given keywords? While the approach based on somewhat homomorphic encryption with BFV [Bra12, FV12] requires communication cost of estimated 314 MB and computation time of about 300 seconds, we estimate that with our HSS this task can be solved with only about a third of the communication (roughly 107 MB) and two orders of magnitude less computation (only about 2.5 seconds). The reason is that the arithmetic circuit computing this function has a multiplicative depth of $\lceil \log(128 \cdot 10 \cdot 4) \rceil = 13$ and thus the parameters of the somewhat homomorphic encryption scheme have to be chosen accordingly larger to account for the noise growth. For our scheme, on the other hand, the size of the parameters depend on the program magnitude only. As all multiplications can be performed over bits, this allows us to choose smaller modulus and ring dimension. Additionally, our multiplication is a comparatively cheap distributed decryption instead of a ciphertext multiplication. For a choice of the underlying parameters we refer to Figure 1.6 (taken with modifications from [BKS19]).

Pseudorandom Correlation Generators. A different approach to secure multiparty computation is taken in protocols in the *preprocessing model*. Here, in an *input-independent preprocessing phase* correlated random strings are generated, which are then used to implement a very efficient *online* phase. An example for a useful correlation are *Beaver triples*, that is, additive secret shares of tuples (a, b, ab). In the online phase Beaver triples allow to efficiently multiply values that are additively secret shared between the parties. The idea is that a and b can serve as one-time-pads to "encrypt" real input values x and y. As this strategy requires one Beaver triple per multiplication, during preprocessing *many* of such (random) triples have to be generated. For this reason the preprocessing phase in such protocols is typically slow and requires a huge amount of communication. Further, saving the correlated random strings takes up a large space in memory for each of the parties.

In the case of secure communication, the problem of exchanging and saving long random strings to encrypt long messages can be solved with a pseudorandom generator: Given a short random seed, the parties can each locally expand this seed to obtain a long pseudorandom string, which can then be used as a one-time pad to encrypt long messages.

The notion of a *pseudorandom correlation generator (PCG)* transfers this idea

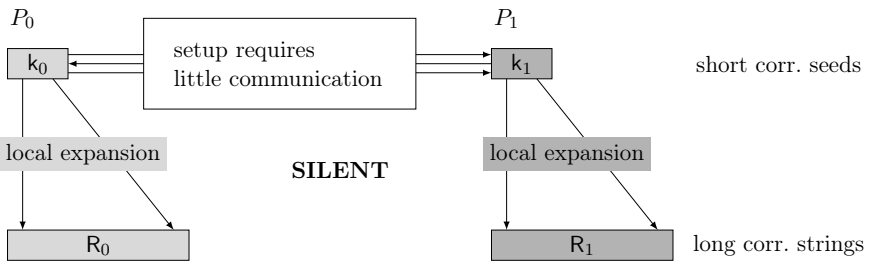

Figure 1.7: How to use pseudorandom correlation generators for *silent* preprocessing. Parties that might want to engage in a protocol for secure computation in the future can engage in a protocol for setting up the short seeds. Once the parties know that a computation is going to take place soon, they can expand their short seeds without further communication. The expansion can be followed by an *efficient* online phase once the inputs are known, where the generated correlated randomness is consumed.

to the more general setting of secure computation: Given short *correlated* seeds, the parties can each expand their seed into long *correlated* pseudorandom strings, e.g. for the correlation of Beaver triples. Replacing the preprocessing phase in a protocol for secure computation comes with two advantages: First, setting up the short correlated seeds requires significantly less communication. Second, the parties only have to save short seeds, which can be silently expanded to long pseudorandom strings whenever the parties want to engage in a secure computation. For a depiction of the preprocessing phase based on pseudorandom correlation generators we refer to Figure 1.7.[14]

In our work [BCG+19b], we advance the state-of-the art both theoretically and practically: Building on a result of [GI99] we show that, unfortunately, pseudorandom correlation generators can not replace correlated randomness in *all* applications. And, even more, that there exist protocols not only for randomized but also for deterministic functionalities that become insecure when instantiated with pseudorandom correlation generators. The reason is, roughly, that the short seed gives an "explanation" of the long string that can not be computed efficiently given the long pseudorandom string only (as this would violate pseudorandomness).

We circumvent this impossibility by giving an *indistinguishability-based* security definition of pseudorandom correlation generators, and show that this definition suffices to replace the preprocessing phase in all protocols for secure computation that satisfy a slightly stronger security notion. Fortunately, natural protocols for secure computation already satisfy this notion, which allows to plug-in PCGs directly in a wide number of applications. Further, we show connection between pseudorandom correlation generators and homomorphic secret sharing schemes.

On the applied side, we give efficient instantiations for a number of useful correlations in the context of secure computation, such as oblivious transfer, one-time correlated truth tables, general low-degree polynomials and Beaver triples. For an overview we refer to Figure 1.8.[15]

In this thesis we focus on the definition of pseudorandom correlation generators,

[14] Figure taken (with modifications) from a presentation by the author on "Circumventing Lower Bounds: Efficient 2-Round OT Extension", Bertinoro Workshop on Lower Bounds, July 2019.
[15] This overview is taken with modifications from [BCG+19b].

Correlation	assumption	efficiency
OT	LPN	1M OT/s
OTTT	PRG	-
deg-d	LPN	-
degree-$d/2$	LPN/MQ + deg-d HSS	-
deg-2 over small ring	LPN+ SXDH	5 OLE/s
deg-d	MQ+ RLWE	6000 ABT/s
multiparty VOLE	LPN	-
multiparty OT/ Beaver triple	LPN+ SXDH/MQ+ RLWE	-

Figure 1.8: Overview of new PCG constructions in [BCG+19b]. OT stands for random oblivious transfer, OTTT for authenticated one-time truth-table correlation, OLE for oblivious linear evaluation (over a constant size ring) and ABT for authenticated Beaver triples. LPN stands for learning parity with noise, PRG for an arbitrary pseudorandom generator, MQ for multivariate quadratic, SXDH for symmetric external Diffie-Hellman and RLWE for learning with errors over rings. Efficiency gives the (approximate) cost over one core of a standard laptop.

present a generic construction of pseudorandom correlation generators from a homomorphic secret sharing scheme together with a suitable pseudorandom generator, and show how to instantiate this pseudorandom correlation generator from lattices for the correlation of authenticated Beaver triples. Authenticated Beaver triples are Beaver triples which are additionally authenticated with a message authentication code. This allows to implement protocols for secure computation even in the presence of parties that deviate from the protocol specification. An example for such a protocol is the so-called SPDZ-protocol [DPSZ12]. Here, the final result is rejected unless the corresponding MAC (for which the key is shared between the parties) verifies. This guarantees correct execution, unless a party guessed the MAC key correctly. In the following we give an overview of the high-level ideas involved.

Defining Pseudorandom Correlation Generators. Typically, security proofs for multi-party computation protocols follow the simulation paradigm (introduced in [GMW87]): The desired behavior is modeled as an *ideal functionality*, and a protocol is said to be *secure* if for all efficient real-world adversaries there exists an efficient simulator who can reproduce the attack in the ideal world, that is, only interacting with the ideal functionality. In particular, everything a dishonest party can learn during the protocol execution must be explicitly modeled by the ideal functionality.

As we want to use PCGs as a plug-in replacement in protocols for secure computation, it would be natural to give a *simulation-based* definition, requiring that in *any* secure protocol execution the use of long correlated strings can be replaced by short correlated PCG seeds. Unfortunately, as sketched in [GI99], this notion turns out to be too strong to be realizable in a non-trivial way: As a counter-example consider the simple protocol, where one party samples a pair of long correlated strings $(\mathsf{R}_0, \mathsf{R}_1)$ and sends R_1 to the other party, who outputs R_1. Replacing the correlated strings by short correlated seeds creates the following problem for the simulator: Given only the output R_1 of the protocol, a simulator reproducing a real transcript would have to efficiently generate a short seed k_1 that can be expanded to R_1. Now, if the PCG construction is non-trivial (i.e. the entropy of a uniformly sampled R_1 exceeds the length of the key k_1), this contradicts the pseudorandomness of R_1.

We thus have to fall back into an indistinguishability-based notion. Intuitively, we require that the correlated pseudorandom strings "look like" real correlated strings (correctness) and the seeds do not leak "too much" about the others party output (security). More precisely, we model the first condition as being *computationally indistinguishable* from correlated strings chosen uniformly at random. For security, we require that a party cannot learn more from their short seed about the others party output, than they can trivially derive from their own output. We show that this security notion is sufficient to directly apply PCGs to a large number of protocols, including [BDOZ11, DPSZ12] based on preprocessed (authenticated) Beaver triples.

Constructing Pseudorandom Correlation Generators. In order to instantiate PCGs for additive correlations, that is, correlations where $\mathsf{R}_0, \mathsf{R}_1$ are subject to $\mathsf{R}_0 + \mathsf{R}_1 = f(X)$ for some function f and input X, we give a generic construction by combining a HSS scheme with an "HSS-friendly" pseudorandom generator PRG expanding a short seed k to a long pseudorandom string X. The idea is as follows: Given a homomorphic secret sharing scheme for the function $f \circ \mathsf{PRG}$, key generation chooses a short k at random, and shares the key into HSS shares k_0 and k_1. Now, for expansion the parties homomorphically evaluate $f \circ \mathsf{PRG}$ on their respective key share. By correctness of the HSS, the outputs add up to $f(X)$ for $X = \mathsf{PRG}(\mathsf{k})$ as required, and, by security, the parties learn nothing about the others party share apart from the obvious. Note that the actual challenge lies in instantiating this approach *efficiently*, as all known efficient HSS constructions only support a limited class of functions. For this reason it is challenging to find a suitable pseudorandom generator. Note that an obvious choice like the low-degree PRG by Goldreich [Gol00, MST03] in fact yields a very inefficient instantiation (see [CDM+18]).

For generating additive shares of authenticated Beaver triples, we instantiate the generic construction with a lattice-based homomorphic secret sharing scheme and a pseudorandom generator based on the *multivariate quadratic (MQ)* assumption. The MQ assumption states, roughly, that evaluating a system of $m > n$ quadratic equations in n variables over a finite field $\mathbb{F}$ on a random point looks pseudorandom. Note that using an MQ-based PRG limits the stretch to subquadratic, as in case $m \geq n^2$ the system of quadratic equations can be solved efficiently via linearization [KPG99]. The reason we still use this PRG is to achieve reasonable computational efficiency.

Regarding the HSS, we use a hybrid scheme based on somewhat homomorphic encryption and our construction from [BKS19] for the most efficient instantiation. This allows to stretch about 3 GB of key material to a total of about 17 GB of authenticated Beaver triples over a large finite field at an estimated rate of more than 6000 authenticated Beaver triples per second.

Related Work. The underlying framework of our homomorphic secret sharing scheme is similar to the construction by Boyle et al. [BGI16a]: *Input values* are represented by encryptions, memory values are represented by secret shares and it is only possible to multiply an *input value* with a *memory value*. Remarkably, compared to the work of Boyle et al. our share-conversion which is necessary to apply after each multiplication is conceptually simpler and much more efficient. Further, it comes with negligible correctness error, and allows to support even superpolynomial input space (by choosing r and q accordingly).

Pseudorandom correlation generators (PCGs) for very simple correlations (that

is, multi-party linear correlations) were first discussed by Gilboa and Ishai in 1999 [GI99]. After the birth of this primitive though there was no progress for almost 20 years—which can be explained possibly by the lack of the right cryptographic tools at the time or the belief that efficient pseudorandom generators are a building block "too good to be true".

Constructions by [HIJ+16, BCG+17, Sch18] for more complex correlations followed, but the first *efficient* pseudorandom correlation generator was given by Boyle et al. [BCGI18] for the correlation of vector oblivious linear evaluation, where a receiver learns a linear combination (of its own choice) of two vectors hold by the sender.

Open Questions. A major open question is constructing efficient homomorphic secret sharing schemes going beyond 2 parties. As described by Benhamouda et al. in [BDIR18], there is an inherent barrier going from 2 to 3 parties for the scheme of [BGI16a]. Also for our construction, this seems to be challenging: Both, the *rounding* and *lifting* trick do not carry over to the 3-party setting. In fact, either would lead to a *constant* correctness error for 3 or more parties.

Another interesting question regards the ciphertext size: Can one get a HSS from lattices without FHE *with polynomial modulus*? Again, our techniques seem not applicable, because they would come with a noticeable correctness error in this setting.

As the research towards efficient pseudorandom correlation generators is relatively young, there is a huge number of questions yet to answer. A dream goal would be to construct a true efficient pseudorandom correlation *function* with exponential (or at least large polynomial) stretch, where correlated randomness can be generated *iteratively*, whenever needed. This would allow parties to communicate securely without further preprocessing *for life time*.

List of Publications

[BCG+19a] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, Peter Rindal and Peter Scholl. **Efficient Two-Round OT Extension and Silent Non-Interactive Secure Computation.** In: *ACM CCS 2019. ACM Press, 2019.*

[BCG+19b] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl. **Efficient Pseudorandom Correlation Generators: Silent OT Extension and More.** In: *CRYPTO 2019. LNCS, vol 11694. Springer, Cham.*

[BKS19] Elette Boyle, Lisa Kohl, and Peter Scholl. **Homomorphic Secret Sharing from Lattices Without FHE.** In: *EUROCRYPT 2019. LNCS, vol 11477. Springer, Cham.*

[Koh19] Lisa Kohl. **Hunting and Gathering - Verifiable Random Functions from Standard Assumptions with Short Proofs.** In: *Public-Key Cryptography – PKC 2019. LNCS, vol 11443. Springer, Cham.*

[HHK18] Julia Hesse, Dennis Hofheinz, Lisa Kohl. **On Tightly Secure Non-Interactive Key Exchange.** In: *CRYPTO 2018. LNCS, vol 10992. Springer, Cham.*

[GHKP18] Romain Gay, Dennis Hofheinz, Lisa Kohl, Jiaxin Pan. **More Efficient (Almost) Tightly Secure Structure-Preserving Signatures.** In: *EUROCRYPT 2018. LNCS, vol 10821. Springer, Cham.*

[GHK17] Romain Gay, Dennis Hofheinz, Lisa Kohl. **Kurosawa-Desmedt Meets Tight Security.** In: *CRYPTO 2017. LNCS, vol 10403. Springer, Cham.*

Chapter 2

Preliminaries

In this chapter we introduce basic cryptographic concepts and notation used throughout this thesis. In particular, we explain the two main cryptographic assumptions, with groups and lattices as underlying mathematical structure, respectively: In the first part of this thesis we work in prime-order groups, where given a group element it is assumed to be hard to compute the discrete logarithm respective to a given generator. We require even more, namely that it is hard to distinguish the product in the exponent from a truly random group element, given the factors in the exponent only. This is the classical and well-studied decisional Diffie-Hellman assumption (or, generalizing to higher dimensions, the matrix Diffie-Hellman assumption).

The second part of the thesis, on the other hand, builds on the more recent assumption that it is hard to solve certain problems in lattices, with the advantage of conjectured security even in the presence of quantum computers. For efficiency we will work with *ideal lattices*, which are a generalization of cyclic lattices. The corresponding assumption is the *learning with errors over rings* assumption.

Subsequently, we introduce basic security notions for hash functions, pseudorandom generators, public-key encryption schemes, signatures and homomorphic secret sharing schemes. For more specialized concepts, we give the preliminaries in the chapter itself.

Note that the preliminaries are in large parts taken verbatim from our works [GHK17], [GHKP18], [BKS19] and [BCG+19b].

Basic Notation. By $\lambda \in \mathbb{N}$ we denote the security parameter. We say a function $\mathsf{negl}\colon \mathbb{N} \to \mathbb{R}_{\geq 0}$ is *negligible*, if for all polynomials $\mathsf{poly} \in \mathbb{N}[X]$ there exists an $n_0 \in \mathbb{N}$ such that $\mathsf{negl}(n) < 1/\mathsf{poly}(n)$ for all $n \geq n_0$. Throughout we consider all parameters to implicitly depend on λ, e.g. by $\ell \in \mathbb{N}$ we actually consider ℓ to be a function $\ell\colon \mathbb{N} \to \mathbb{N}$, but simply write ℓ in order to refer to $\ell(\lambda)$. We say $\ell \leq \mathsf{poly}(\lambda)$, if there exists a polynomial $p[X] \in \mathbb{N}[X]$ and a $\lambda_0 \in \mathbb{N}$ such that for all $\lambda \geq \lambda_0$ we have $\ell(\lambda) \leq \mathsf{poly}(\lambda)$. If no such polynomial exist, we write $\ell \geq \lambda^{\omega(1)}$.

For an arbitrary set $\mathcal{S}$, by $x \leftarrow_R \mathcal{S}$ we denote the process of sampling an element x from $\mathcal{S}$ uniformly at random. For any distribution $\mathcal{D}$ by $d \leftarrow \mathcal{D}$ we denote the process of sampling an element d according to the distribution $\mathcal{D}$. We say that $\mathcal{P}$ is *probabilistic polynomial time* (PPT), if $\mathcal{P}$ is a probabilistic algorithm with running time polynomial in λ. We use $y \leftarrow \mathcal{P}(x)$ to denote that y is assigned the output of $\mathcal{P}$ running on input x. For a deterministic algorithm we sometimes use the notation $y := \mathcal{P}(x)$ instead. We generalize $\mathcal{P}$ to vectors of inputs in a straightforward way: $\mathcal{P}$ is run independently on each entry of the vector (with independent random coins if $\mathcal{P}$ is randomized).

For any bit string $\tau \in \{0,1\}^*$, we denote by τ_i the i-th bit of τ and by $\tau_{|i} \in \{0,1\}^i$ the bit string comprising the first i bits of τ. Similarly, for any element $m \in \mathbb{Z}_p$ (for

some $p \in \mathbb{N}$), we denote by $m_i \in \{0,1\}$ the i-th bit of m's bit representation and by $m_{|i} \in \{0,1\}^i$ the bit string comprising the first i bits of m's bit representation. For $\ell \in \mathbb{N}$ and p a prime, by $\mathbb{F}_{p^\ell}$ we denote a finite field consisting of p^ℓ elements. For a real number $x \in \mathbb{R}$, by $\lfloor x \rceil \in \mathbb{Z}$ we denote the element closest to $x \in \mathbb{R}$, where we round up when the first decimal place of x is 5 or higher.

Vector and Matrix Notation. We denote vectors by bold lower-case letters and matrices by bold upper-case letters. We interpret vectors as column-vectors. For a vector $\mathbf{x} \in R^\ell$, by x_i we refer to the i-th entry (for $i \in \{1, \ldots, \ell\}$).

Let p be a prime. Let $k, \ell \in \mathbb{N}$ such that $\ell > k$. Then for any matrix $\mathbf{A} \in \mathbb{Z}_p^{\ell \times k}$, we write $\overline{\mathbf{A}} \in \mathbb{Z}_p^{k \times k}$ for the upper square matrix of $\mathbf{A}$, and $\underline{\mathbf{A}} \in \mathbb{Z}_p^{(\ell-k) \times k}$ for the lower $\ell - k$ rows of $\mathbf{A}$. With

$$\mathrm{span}(\mathbf{A}) := \{\mathbf{A}\mathbf{r} \mid \mathbf{r} \in \mathbb{Z}_p^k\} \subset \mathbb{Z}_p^\ell,$$

we denote the *span* of $\mathbf{A}$.

For vectors $\mathbf{v} \in \mathbb{Z}_p^{2k}$, by $\overline{\mathbf{v}} \in \mathbb{Z}_p^k$ we denote the vector consisting of the upper k entries of $\mathbf{v}$ and accordingly by $\underline{\mathbf{v}} \in \mathbb{Z}_p^k$ we denote the vector consisting of the lower k entries of $\mathbf{v}$.

As usual by $\mathbf{A}^\top \in \mathbb{Z}_p^{k \times \ell}$ we denote the *transpose* of $\mathbf{A}$ and if $\ell = k$ and $\mathbf{A}$ is invertible by $\mathbf{A}^{-1} \in \mathbb{Z}_p^{\ell \times \ell}$ we denote the *inverse* of $\mathbf{A}$.

For $\ell \geq k$ by $\mathbf{A}^\perp$ we denote a matrix in $\mathbb{Z}_p^{\ell \times (\ell-k)}$ with $\mathbf{A}^\top \mathbf{A}^\perp = \mathbf{0}$ and rank $\ell - k$. We denote the set of all matrices with these properties as

$$\mathrm{orth}(\mathbf{A}) := \{\mathbf{A}^\perp \in \mathbb{Z}_p^{\ell \times (\ell-k)} \mid \mathbf{A}^\top \mathbf{A}^\perp = \mathbf{0} \text{ and } \mathbf{A}^\perp \text{ has rank } \ell - k\}.$$

Finally, for $l = k$ we denote the *trace* of $\mathbf{A}$, that is the sum of the diagonal elements of $\mathbf{A}$, by

$$\mathrm{trace}(\mathbf{A}) := \sum_{i=1}^{k} \mathbf{A}_{i,i}.$$

2.1 Cryptography from Groups

Let GGen be a PPT algorithm that on input 1^λ returns a description $\mathcal{G} = (\mathbb{G}, p, P)$ of an additive cyclic group $\mathbb{G}$ of order p for a 2λ-bit prime p, whose generator is P.

We use the representation of group elements introduced in [EHK+13]. Namely, for $a \in \mathbb{Z}_p$, define $[a] = aP \in \mathbb{G}$ as the *implicit representation* of a in $\mathbb{G}$. More generally, for a matrix $\mathbf{A} = (a_{ij}) \in \mathbb{Z}_p^{\ell \times k}$ we define $[\mathbf{A}]$ as the implicit representation of $\mathbf{A}$ in $\mathbb{G}$:

$$[\mathbf{A}] := \begin{pmatrix} a_{11}P & \ldots & a_{1k}P \\ & & \\ a_{\ell 1}P & \ldots & a_{\ell k}P \end{pmatrix} \in \mathbb{G}^{\ell \times k}$$

Note that from $[a] \in \mathbb{G}$ it is hard to compute the value a if the discrete logarithm assumption holds in $\mathbb{G}$. Obviously, given $[a], [b] \in \mathbb{G}$ and a scalar $x \in \mathbb{Z}_p$, one can efficiently compute $[ax] \in \mathbb{G}$ and $[a + b] \in \mathbb{G}$.

For matrices $\mathbf{A} \in \mathbb{Z}_p^{l \times k}, \mathbf{B} \in \mathbb{Z}_p^{l \times k}$, by $[\mathbf{A}, \mathbf{B}]$ we denote the composed matrix $\begin{bmatrix} \mathbf{A} \\ \mathbf{B} \end{bmatrix} \in \mathbb{G}^{2l \times k}$. Further, by

$$\mathrm{span}([\mathbf{A}]) := \{[\mathbf{A}]\mathbf{r} \mid \mathbf{r} \in \mathbb{Z}_p^k\} \subset \mathbb{G}^l$$

we denote the span of $[\mathbf{A}]$ in $\mathbb{G}^l$ and by

$$\mathrm{trace}([\mathbf{A}]) := \left[\sum_{i=1}^{k} \mathbf{A}_{i,i}\right]$$

the trace of $[\mathbf{A}]$ in $\mathbb{G}$.

Let BGen be a probabilistic polynomial time (PPT) algorithm that on input 1^λ returns a description $\mathcal{PG} = (\mathbb{G}_1, \mathbb{G}_2, G_T, p, P_1, P_2, e)$ of asymmetric pairing groups where $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$ are cyclic group of order p for a 2λ-bit prime p, P_1 and P_2 are generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively, and $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is an efficiently computable (non-degenerate) bilinear map. Define $P_T := e(P_1, P_2)$, which is a generator of $\mathbb{G}_T$. Again, we use implicit representation of group elements. For $i \in \{1, 2, T\}$ and $a \in \mathbb{Z}_p$, we define $[a]_i = aP_i \in \mathbb{G}_i$ as the implicit representation of a in $\mathbb{G}_i$. Given $[a]_1$, $[a]_2$, one can efficiently compute $[ab]_T$ using the pairing e. For two matrices $\mathbf{A}$, $\mathbf{B}$ with matching dimensions, we define $e([\mathbf{A}]_1, [\mathbf{B}]_2) := [\mathbf{AB}]_T \in \mathbb{G}_T$.

We recall the definitions of the Matrix Decision Diffie-Hellman (MDDH) assumption from [EHK+13].

Definition 1 (Matrix distribution)**.** Let $k, \ell \in \mathbb{N}$, with $\ell > k$ and p be a 2λ-bit prime. We call $\mathcal{D}_{\ell,k}$ a *matrix distribution* if it outputs matrices in $\mathbb{Z}_p^{\ell \times k}$ of full rank k in polynomial time.

In the following we only consider matrix distributions $\mathcal{D}_{\ell,k}$, where for all $\mathbf{A} \leftarrow_R \mathcal{D}_{\ell,k}$ the first k rows of $\mathbf{A}$ form an invertible matrix. We also require that in case $\ell = 2k$ for any two matrices $\mathbf{A}_0, \mathbf{A}_1 \leftarrow_R \mathcal{D}_{2k,k}$ the matrix $(\mathbf{A}_0 \mid \mathbf{A}_1)$ has full rank with overwhelming probability. In the following we will denote this probability by $1 - \Delta_{\mathcal{D}_{2k,k}}$. Note that if $(\mathbf{A}_0 \mid \mathbf{A}_1)$ has full rank, then for any $\mathbf{A}_0^\perp \in \mathrm{orth}(\mathbf{A}_0)$, $\mathbf{A}_1^\perp \in \mathrm{orth}(\mathbf{A}_1)$ the matrix $(\mathbf{A}_0^\perp \mid \mathbf{A}_1^\perp) \in \mathbb{Z}_p^{2k \times 2k}$ has full rank as well, as otherwise there would exists a non-zero vector $\mathbf{v} \in \mathbb{Z}_p^{2k} \backslash \{\mathbf{0}\}$ with $(\mathbf{A}_0 \mid \mathbf{A}_1)^\top \mathbf{v} = \mathbf{0}$. Further, by similar reasoning $(\mathbf{A}_0^\perp)^\top \mathbf{A}_1 \in \mathbb{Z}_p^{k \times k}$ has full rank.

The $\mathcal{D}_{\ell,k}$-Matrix Diffie-Hellman problem is, for a randomly chosen $\mathbf{A} \leftarrow_R \mathcal{D}_{\ell,k}$, to distinguish the between tuples of the form $([\mathbf{A}], [\mathbf{Aw}])$ and $([\mathbf{A}], [\mathbf{u}])$, where $\mathbf{w} \leftarrow_R \mathbb{Z}_p^k$ and $\mathbf{u} \leftarrow_R \mathbb{Z}_p^\ell$.

Definition 2 ($\mathcal{D}_{\ell,k}$-Matrix Diffie-Hellman $\mathcal{D}_{\ell,k}$-MDDH)**.** Let $\mathcal{D}_{\ell,k}$ be a matrix distribution. We say that the *$\mathcal{D}_{\ell,k}$-Matrix Diffie-Hellman ($\mathcal{D}_{\ell,k}$-MDDH) assumption* holds relative to a prime order group $\mathbb{G}$ if for all PPT adversaries $\mathcal{A}$,

$$\begin{aligned}\mathrm{Adv}^{\mathrm{mddh}}_{\mathbb{G},\mathcal{D}_{\ell,k},\mathcal{A}}(\lambda) :&= |\Pr[\mathcal{A}(\mathcal{G}, [\mathbf{A}], [\mathbf{Aw}]) = 1] - \Pr[\mathcal{A}(\mathcal{G}, [\mathbf{A}], [\mathbf{u}]) = 1]| \\ &\leq \mathsf{negl}(\lambda),\end{aligned}$$

where the probabilities are taken over $\mathcal{G} := (\mathbb{G}, p, P) \leftarrow_R \mathsf{GGen}(1^\lambda)$, $\mathbf{A} \leftarrow_R \mathcal{D}_{\ell,k}, \mathbf{w} \leftarrow_R \mathbb{Z}_p^k, \mathbf{u} \leftarrow_R \mathbb{Z}_p^\ell$.

For $Q \in \mathbb{N}$, $\mathbf{W} \leftarrow_R \mathbb{Z}_p^{k \times Q}$ and $\mathbf{U} \leftarrow_R \mathbb{Z}_p^{\ell \times Q}$, we consider the Q-fold $\mathcal{D}_{\ell,k}$-MDDH assumption, which states that distinguishing tuples of the form $([\mathbf{A}], [\mathbf{AW}])$ from $([\mathbf{A}], [\mathbf{U}])$ is hard. That is, a challenge for the Q-fold $\mathcal{D}_{\ell,k}$-MDDH assumption consists of Q independent challenges of the $\mathcal{D}_{\ell,k}$-MDDH Assumption (with the same $\mathbf{A}$ but different randomness $\mathbf{w}$). In [EHK+13] it is shown that the two problems are equivalent, where the reduction loses at most a factor $\ell - k$.

Lemma 3 (Random self-reducibility of $\mathcal{D}_{\ell,k}$-MDDH, [EHK+13])**.** *Let $\ell, k, Q \in \mathbb{N}$ with $\ell > k$ and $Q > \ell - k$. For any PPT adversary $\mathcal{A}$, there exists an adversary $\mathcal{B}$ such that $T(\mathcal{B}) \approx T(\mathcal{A}) + Q \cdot \mathsf{poly}(\lambda)$ with $\mathsf{poly}(\lambda)$ independent of $T(\mathcal{A})$, and*

$$\mathrm{Adv}^{Q\text{-mddh}}_{\mathbb{G},\mathcal{D}_{\ell,k},\mathcal{A}}(\lambda) \leq (\ell - k) \cdot \mathrm{Adv}^{\mathrm{mddh}}_{\mathbb{G},\mathcal{D}_{\ell,k},\mathcal{B}}(\lambda) + \frac{1}{p-1}.$$

Here

$$\mathrm{Adv}^{Q\text{-mddh}}_{\mathbb{G},\mathcal{D}_{\ell,k},\mathcal{A}}(\lambda) := \left|\Pr[\mathcal{A}(\mathcal{G}, [\mathbf{A}], [\mathbf{AW}]) = 1] - \Pr[\mathcal{A}(\mathcal{G}, [\mathbf{A}], [\mathbf{U}]) = 1]\right|,$$

where the probability is over $\mathcal{G} := (\mathbb{G}, p, p) \leftarrow_R \mathsf{GGen}(1^\lambda)$, $\mathbf{A} \leftarrow_R \mathcal{U}_{\ell,k}, \mathbf{W} \leftarrow_R \mathbb{Z}_p^{k \times Q}$ and $\mathbf{U} \leftarrow_R \mathbb{Z}_p^{\ell \times Q}$.

The uniform distribution is a particular matrix distribution that deserves special attention, as an adversary breaking the $\mathcal{U}_{\ell,k}$-MDDH assumption can also distinguish between real MDDH tuples and random tuples for all other possible matrix distributions.

Definition 4 (Uniform distribution)**.** Let $\ell, k \in \mathbb{N}$, with $\ell \geq k$, and a prime p. We denote by $\mathcal{U}_{\ell,k}$ the *uniform distribution* over all full-rank $\ell \times k$ matrices over $\mathbb{Z}_p$. Let $\mathcal{U}_k := \mathcal{U}_{k+1,k}$.

Lemma 5 ($\mathcal{D}_{\ell,k}$-$MDDH \Rightarrow \mathcal{U}_{\ell,k}$-$MDDH$, [EHK+13])**.** *Let $\mathcal{D}_{\ell,k}$ be a matrix distribution. For any adversary $\mathcal{A}$ on the $\mathcal{U}_{\ell,k}$-distribution, there exists an adversary $\mathcal{B}$ on the $\mathcal{D}_{\ell,k}$-assumption such that $T(\mathcal{B}) \approx T(\mathcal{A})$ and $\mathrm{Adv}^{\mathrm{mddh}}_{\mathbb{G},\mathcal{U}_{\ell,k},\mathcal{A}}(\lambda) = \mathrm{Adv}^{\mathrm{mddh}}_{\mathbb{G},\mathcal{D}_{\ell,k},\mathcal{B}}(\lambda)$.*

We state a tighter random-self reducibility property for case of the uniform distribution.

Lemma 6 (Random self-reducibility of $\mathcal{U}_{\ell,k}$-MDDH, [EHK+13])**.** *Let $\ell, k, Q \in \mathbb{N}$ with $\ell > k$. For any PPT adversary $\mathcal{A}$, there exists an adversary $\mathcal{B}$ such that $T(\mathcal{B}) \approx T(\mathcal{A}) + Q \cdot \mathsf{poly}(\lambda)$ with $\mathsf{poly}(\lambda)$ independent of $T(\mathcal{A})$, and*

$$\mathrm{Adv}^{Q\text{-mddh}}_{\mathbb{G},\mathcal{U}_{\ell,k},\mathcal{A}}(\lambda) \leq \mathrm{Adv}^{\mathrm{mddh}}_{\mathbb{G},\mathcal{U}_{\ell,k},\mathcal{B}}(\lambda) + \frac{1}{p-1}.$$

We also recall this property of the uniform distribution, stated in [GHKW16].

Lemma 7 ($\mathcal{U}_k$-MDDH $\Leftrightarrow \mathcal{U}_{\ell,k}$-MDDH)**.** *Let $\ell, k \in \mathbb{N}$, with $\ell > k$. For any adversary $\mathcal{A}$, there exists an adversary $\mathcal{B}$ (and vice versa) such that $T(\mathcal{B}) \approx T(\mathcal{A})$ and $\mathrm{Adv}^{\mathrm{mddh}}_{\mathbb{G},\mathcal{U}_{\ell,k},\mathcal{A}}(\lambda) = \mathrm{Adv}^{\mathrm{mddh}}_{\mathbb{G},\mathcal{U}_k,\mathcal{B}}(\lambda)$.*

For $k \in \mathbb{N}$ we define $\mathcal{D}_k := \mathcal{D}_{k+1,k}$.

The Kernel-Diffie-Hellman assumption $\mathcal{D}_k$-KMDH [MRV16] is a natural *computational analogue* of the $\mathcal{D}_k$-MDDH Assumption.

Definition 8 ($\mathcal{D}_k$-Kernel Diffie-Hellman assumption $\mathcal{D}_k$-KMDH)**.** Let $\mathcal{D}_k$ be a matrix distribution. We say that the $\mathcal{D}_k$-Kernel Diffie-Hellman ($\mathcal{D}_k$-KMDH) assumption holds relative to a prime order group $\mathbb{G}_i$ for $i \in \{1, 2\}$ if for all PPT adversaries $\mathcal{A}$,

$$\begin{aligned}\mathrm{Adv}^{\mathrm{kmdh}}_{\mathcal{PG},\mathbb{G}_i,\mathcal{D}_{\ell,k},\mathcal{A}}(\lambda) :&= \Pr[\mathbf{c}^\top \mathbf{A} = \mathbf{0} \wedge \mathbf{c} \neq \mathbf{0} \mid [\mathbf{c}]_{3-i} \leftarrow_R \mathcal{A}(\mathcal{PG}, [\mathbf{A}]_i)] \\ &\leq \mathsf{negl}(\lambda),\end{aligned}$$

where the probabilities are taken over $\mathcal{PG} := (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, P_1, P_2) \leftarrow \mathsf{BGen}(1^\lambda)$, and $\mathbf{A} \leftarrow_R \mathcal{D}_k$.

Note that we can use a non-zero vector in the kernel of $\mathbf{A}$ to test membership in the column space of $\mathbf{A}$. This means that the $\mathcal{D}_k$-KMDH assumption is a relaxation of the $\mathcal{D}_k$-MDDH assumption, as captured in the following lemma from [MRV16].

Lemma 9 ([MRV16])**.** *For any matrix distribution $\mathcal{D}_k$, $\mathcal{D}_k$-MDDH $\Rightarrow$ $\mathcal{D}_k$-KMDH.*

In this paper, we are particularly interested in the case $k = 1$, which corresponds to the DDH assumption, that we recall here.

Definition 10 (DDH)**.** We say that the DDH assumption holds relative to a prime order group $\mathbb{G}$ if for all PPT adversaries $\mathcal{A}$,

$$\mathrm{Adv}^{\mathrm{ddh}}_{\mathbb{G},\mathcal{A}}(\lambda) := |\Pr[\mathcal{A}(\mathcal{G}, [a], [r], [ar]) = 1] - \Pr[\mathcal{A}(\mathcal{G}, [a], [r], [b]| \leq \mathsf{negl}(\lambda),$$

where the probabilities are taken over $\mathcal{G} := (\mathbb{G}, p, P) \leftarrow_R \mathsf{GGen}(1^\lambda)$, $a, b, r \leftarrow_R \mathbb{Z}_p$.

Note that the DDH assumption is equivalent to $\mathcal{D}_{2,1}$-MDDH, where $\mathcal{D}_{2,1}$ is the distribution that outputs matrices $\binom{1}{a}$, for $a \leftarrow_R \mathbb{Z}_p$ chosen uniformly at random.

For $\mathcal{PG} = (\mathbb{G}_1, \mathbb{G}_2, G_T, p, P_1, P_2, e)$, assuming $\mathcal{D}_{2,1}$-MDDH relative to $\mathbb{G}_1$ and relative to $\mathbb{G}_2$, corresponds to the symmetric external Diffie-Hellman (SXDH) assumption.

2.2 Cryptography from Lattices

A *lattice* is a discrete subgroup of $\mathbb{R}^n$ (for $n \in \mathbb{N}$). Lattice-based cryptography was initiated by the work of Ajtai [Ajt96], who gave the construction of a collision resistant hash function whose hardness can be reduced to solving a lattice problem in the worst case.

The *learning with errors (LWE)* problem, which states that it is hard to distinguish *noisy* linear combinations from truly random ones, was introduced by Regev in 2005 [Reg05], who also showed connections of the LWE assumption to worst-case lattice problems via a quantum reduction. Later, classical reductions followed [Pei09, BLP+13], and since numerous cryptographic building blocks have been constructed whose hardness can be reduced to LWE. Apart from the worst-case hardness, cryptography based on lattices is appealing, as—opposed to cryptography from groups—lattice problems seem to persist quantum attacks and thus give long-term security guarantees.

Definition 11 (Learning With Errors (LWE))**.** Let $d \in \mathsf{poly}(\lambda)$ $q \geq 2$ be an integer, and $\mathbb{Z}_q = \mathbb{Z}/(q\mathbb{Z})$. Let $\mathcal{D}_{\mathrm{err}}$ be an error distribution over $\mathbb{Z}$ and $\mathcal{D}_{\mathsf{sk}}$ be a secret key distribution over $\mathbb{Z}^d$. Let $\mathbf{s} \leftarrow \mathcal{D}_{\mathsf{sk}}$. The $\mathsf{LWE}_{d,q,\mathcal{D}_{\mathrm{err}},\mathcal{D}_{\mathsf{sk}}}$ problem is to distinguish the following two distributions over $\mathbb{Z}_q^d \times \mathbb{Z}_q$:

- $\mathcal{O}_{\mathcal{D}_{\mathrm{err}},\mathbf{s}}$: Output $(\mathbf{a}, b)$ where $\mathbf{a} \leftarrow \mathbb{Z}_q^d, e \leftarrow \mathcal{D}_{\mathrm{err}}$ and $b = \langle \mathbf{a}, \mathbf{s} \rangle + e \mod q$
- U: Output $(\mathbf{a}, u) \leftarrow \mathbb{Z}_q^d \times \mathbb{Z}_q$

Formally, for a PPT adversary $\mathcal{A}$ we define the advantage

$$\mathrm{Adv}^{\mathrm{lwe}}_{d,q,\mathcal{D}_{\mathrm{err}},\mathcal{D}_{\mathsf{sk}}}(\lambda) = |\Pr_{\mathbf{s}\leftarrow\mathcal{D}_{\mathsf{sk}}}[\mathcal{A}^{\mathcal{O}_{\mathcal{D}_{\mathrm{err}},\mathbf{s}}}(\lambda) = 1] - \Pr_{\mathbf{s}\leftarrow\mathcal{D}_{\mathsf{sk}}}[\mathcal{A}^{U}(\lambda) = 1]|.$$

Additionally, we work with ideal lattices, where the additional algebraic structure can be exploited for more efficient instantiations [LPR13]. More precisely, throughout our underlying ring will be of the form $R := \mathbb{Z}[X]/(X^N + 1)$ for $N \in \mathsf{poly}(\lambda)$ a power of 2. Note that we restrict N to powers of 2 to keep the analysis simpler.

Definition 12 (Learning With Errors over Rings (RLWE))**.** Let $N \in \mathsf{poly}(\lambda)$ be a power of 2, $q \geq 2$ be an integer, $R = \mathbb{Z}[X]/(X^N + 1)$ and $R_q = R/(qR)$. Let $\mathcal{D}_{\mathrm{err}}$ be an error distribution over R and $\mathcal{D}_{\mathsf{sk}}$ be a secret key distribution over R. Let $s \leftarrow \mathcal{D}_{\mathsf{sk}}$. The $\mathsf{RLWE}_{N,q,\mathcal{D}_{\mathrm{err}},\mathcal{D}_{\mathsf{sk}}}$ problem is to distinguish the following two distributions over R_q^2:

- $\mathcal{O}_{\mathcal{D}_{\mathrm{err}},s}$: Output (a, b) where $a \leftarrow R_q, e \leftarrow \mathcal{D}_{\mathrm{err}}$ and $b = a \cdot s + e \mod q$.
- U: Output $(a, u) \leftarrow R_q^2$.

Formally, for a PPT adversary $\mathcal{A}$ we define the advantage

$$\mathrm{Adv}^{\mathrm{rlwe}}_{N,q,\mathcal{D}_{\mathrm{err}},\mathcal{D}_{\mathsf{sk}}}(\lambda) = |\Pr_{s\leftarrow\mathcal{D}_{\mathsf{sk}}}[\mathcal{A}^{\mathcal{O}_{\mathcal{D}_{\mathrm{err}},s}}(\lambda) = 1] - \Pr_{s\leftarrow\mathcal{D}_{\mathsf{sk}}}[\mathcal{A}^{U}(\lambda) = 1]|.$$

If $\mathcal{D}_{\mathrm{err}} = \mathcal{D}_{\mathsf{sk}}$, we simply write $\mathsf{RLWE}_{N,q,\mathcal{D}_{\mathrm{err}}}$.

For $x \in R$ the maximum norm of x is defined as $\|x\|_\infty := \max_{i=0}^{N-1} |x_i|$, where $x_i \in \mathbb{Z}$ such $x = \sum_{i=0}^{N-1} x_i X^i \mod X^N + 1$. For $B \in \mathbb{N}$, we denote $[R]_B := \{x \in R \mid \|x\|_\infty \leq B\}$. More generally, for an interval $I \subseteq \mathbb{Z}$, we write $R|_I$ to denote all elements of R that have only coefficients in I.

For $r \in \mathbb{N}$, by R_r we denote $R/(rR)$. Note that we consider R_r as elements for which all coefficients are in the interval $(-\lfloor r/2 \rceil, \ldots, \lfloor (r-1)/2 \rceil]$.

2.3 Basic Cryptographic Building Blocks

A hash function generator is a probabilistic polynomial time algorithm $\mathcal{H}$ that, on input 1^λ, outputs an efficiently computable function $\mathsf{H} : \{0,1\}^* \to \{0,1\}^\lambda$, unless domain and co-domain are explicitly specified.

Definition 13 (Collision resistance)**.** We say that a hash function generator $\mathcal{H}$ outputs *collision resistant functions* H, if for all PPT adversaries $\mathcal{A}$ and $\mathsf{H} \leftarrow_R \mathcal{H}(1^\lambda)$ it holds

$$\mathrm{Adv}^{\mathrm{CR}}_{\mathcal{H},\mathcal{A}}(\lambda) := \Pr\left[x \neq x' \wedge \mathsf{H}(x) = \mathsf{H}(x') \mid (x, x') \leftarrow \mathcal{A}(1^\lambda, \mathsf{H})\right] \leq \mathsf{negl}(\lambda).$$

We say a hash function is *collision resistant* if it is sampled from a collision resistant hash function generator.

Definition 14 (Universality)**.** We say a hash function generator $\mathcal{H}$ is *universal*, if for every $x, x' \in \{0,1\}^*$ with $x \neq x'$ it holds

$$\Pr\left[\mathsf{h}(x) = \mathsf{h}(x') \mid \mathsf{h} \leftarrow_R \mathcal{H}(1^\lambda)\right] = \frac{1}{2^\lambda}.$$

We say a hash function is *universal* if it is sampled from a universal hash function generator.

Lemma 15 (Leftover Hash Lemma [ILL89])**.** *Let $\mathcal{X}, \mathcal{Y}$ be sets, $\ell \in \mathbb{N}$ and $\mathsf{h}\colon \mathcal{X} \to \mathcal{Y}$ be a universal hash function. Then for all $X \leftarrow_R \mathcal{X}$, $U \leftarrow_R \mathcal{Y}$ and $\varepsilon > 0$ with $\log|\mathcal{X}| \geq \log|\mathcal{Y}| + 2\log\varepsilon$ we have*

$$\Delta\left((\mathsf{h}, \mathsf{h}(X)), (\mathsf{h}, U)\right) \leq \frac{1}{\varepsilon},$$

where Δ denotes the statistical distance.

Another very basic cryptographic building block that we use for constructing pseudorandom correlation generators is a pseudorandom generator.

Definition 16 (Pseudorandom Generator)**.** Let $\mathcal{X}, \mathcal{Y}$ be sets. We say $\mathsf{PRG}\colon \mathcal{X} \to \mathcal{Y}$ is a *pseudorandom generator (PRG)*, if for all PPT adversaries $\mathcal{A}$ the advantage

$$\mathrm{Adv}^{\mathrm{prg}}_{\mathsf{PRG},\mathcal{A}} := \left|\Pr[\mathcal{A}(1^\lambda, \mathsf{PRG}(X)) = 1 \mid X \leftarrow_R \mathcal{X}] - \Pr[\mathcal{A}(1^\lambda, Y) = 1 \mid Y \leftarrow_R \mathcal{Y}]\right|$$

is negligible in λ.

2.4 Public-Key Encryption

Definition 17 (Public-key encryption)**.** A *public-key encryption scheme* is a tuple of three PPT algorithms $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ such that:

$\mathsf{Gen}(1^\lambda)$: On input of the security parameter λ in unary representation, returns a pair $(\mathsf{pk}, \mathsf{sk})$ of a public and a secret key.

$\mathsf{Enc}(\mathsf{pk}, M)$: On input of the public key pk and a message M, returns a ciphertext C.

$\mathsf{Dec}(\mathsf{sk}, C)$: On input of the secret key sk and ciphertext C, returns a message M or a special rejection symbol $\perp$.

We say $\mathsf{PKE} := (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is *perfectly correct*, if for all $\lambda \in \mathbb{N}$,

$$\Pr[\mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, M)) = M] = 1,$$

where the probability is over $(\mathsf{pk}, \mathsf{sk}) \leftarrow_R \mathsf{Gen}(1^\lambda)$, $C \leftarrow_R \mathsf{Enc}(\mathsf{pk}, M)$.

Note that we always implicitly assume the secret key to contain the public key.

Definition 18 (Multi-ciphertext CCA security)**.** For any public-key encryption scheme $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ and any stateful adversary $\mathcal{A}$, we define the following security experiment:

$\underline{\mathrm{Exp}^{\mathrm{cca}}_{\mathsf{PKE},\mathcal{A}}(\lambda)}$:	$\underline{\mathcal{O}_{\mathrm{enc}}(M_0, M_1)}$:	$\underline{\mathcal{O}_{\mathrm{dec}}(C)}$:
$(pk, sk) \leftarrow_R \mathsf{Gen}(1^\lambda)$	**if** $\lvert M_0\rvert = \lvert M_1\rvert$	**if** $C \notin \mathcal{C}_{\mathbf{enc}}$
$b \leftarrow_R \{0,1\}$	$C \leftarrow_R \mathsf{Enc}(\mathsf{pk}, M_b)$	$M := \mathsf{Dec}(\mathsf{sk}, C)$
$\mathcal{C}_{\mathbf{enc}} := \emptyset$	$\mathcal{C}_{\mathbf{enc}} := \mathcal{C}_{\mathbf{enc}} \cup \{C\}$	**return** M
$b' \leftarrow_R \mathcal{A}^{\mathcal{O}_{\mathrm{enc}}(\cdot,\cdot),\mathcal{O}_{\mathrm{dec}}(\cdot)}(\mathsf{pk})$	**return** C	**else return** $\perp$
if $b = b'$ **return** 1		
else return 0		

$\underline{\mathrm{Exp}^{\mathrm{pr}}_{\mathsf{PKE},\mathcal{A}}(\lambda):}$	$\underline{\mathcal{O}_{\mathrm{enc}}(m):}$
$(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{PKE.Gen}(1^\lambda)$	**if** $\beta = 0$
$\beta \leftarrow \{0,1\}$	$\quad c \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}, m)$
$\beta' \leftarrow \mathcal{A}^{\mathcal{O}_{\mathrm{enc}}(\cdot)}(1^\lambda, \mathsf{pk})$	**return** c
if $\beta = \beta'$ **return** 1	**else**
else return 0	$\quad c \leftarrow_R \mathcal{C}$
	$\quad$ **return** c

Figure 2.1: Security challenge experiment for pseudorandomness of ciphertexts.

We say PKE is IND-CCA secure, if for all PPT adversaries $\mathcal{A}$, the advantage

$$\mathrm{Adv}^{\mathrm{cca}}_{\mathsf{PKE},\mathcal{A}}(\lambda) := \left|\Pr[\mathrm{Exp}^{\mathrm{cca}}_{\mathsf{PKE},\mathcal{A}}(\lambda) = 1] - \frac{1}{2}\right| \leq \mathsf{negl}(\lambda).$$

In our work [BKS19], we use a different security notion. Namely, we require that ciphertexts look *pseudorandom*.

Definition 19 (Pseudorandomness of ciphertexts)**.** We say a public-key encryption scheme $\mathsf{PKE} := (\mathsf{PKE.Gen}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ with ciphertext space $\mathcal{C}$ satisfies *pseudorandomness of ciphertexts* or simply PKE is *secure*, if for every PPT adversary $\mathcal{A}$ the advantage

$$\mathrm{Adv}^{\mathrm{pr}}_{\mathsf{PKE},\mathcal{A}}(\lambda) := \left|\Pr\left[\mathrm{Exp}^{\mathrm{pr}}_{\mathsf{PKE},\mathcal{A}}(\lambda) = 1\right] - \frac{1}{2}\right|$$

is negligible in λ, where $\mathrm{Exp}^{\mathrm{pr}}_{\mathsf{PKE},\mathcal{A}}(\lambda)$ is as defined in Figure 2.1.

2.5 Secret-Key Encryption

In a secret-key encryption scheme, the key for en- and decryption is the same.

Definition 20 (Secret-key encryption)**.** A *secret-key encryption scheme* is a tuple of three PPT algorithms $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ such that:

$\mathsf{Gen}(1^\lambda)$: On input of the security parameter λ in unary representation, returns a secret key sk.

$\mathsf{Enc}(\mathsf{sk}, M)$: On input of the secret key sk and a message M, returns a ciphertext C.

$\mathsf{Dec}(\mathsf{sk}, C)$: On input of the secret key sk and ciphertext C, returns a message M or a special rejection symbol $\perp$.

We say $\mathsf{PKE} := (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is *perfectly correct*, if for all $\lambda \in \mathbb{N}$,

$$\Pr[\mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{sk}, M)) = M] = 1,$$

where the probability is over $\mathsf{sk} \leftarrow_R \mathsf{Gen}(1^\lambda)$, $C \leftarrow_R \ \mathsf{Enc}(\mathsf{pk}, M)$.

For the secret-key version of our homomorphic secret sharing in [BKS19], the underlying encryption scheme has to satisfy KDM-security defined in the following.

$\underline{\mathrm{Exp}^{\mathrm{kdm}}_{\mathsf{SKE},\Upsilon,\mathcal{A}}(\lambda):}$	$\underline{\mathcal{O}_{\mathrm{enc}}(f):}$
$\mathsf{sk} \leftarrow \mathsf{PKE.Gen}(1^\lambda)$	**if** $\beta = 0$
$\beta \leftarrow \{0,1\}$	$\quad c \leftarrow \mathsf{PKE.Enc}(\mathsf{sk}, f(\mathsf{sk}))$
$\beta' \leftarrow \mathcal{A}^{\mathcal{O}_{\mathrm{enc}}(\cdot)}(1^\lambda)$	**return** c
if $\beta = \beta'$ **return** 1	**else**
else return 0	$\quad c \leftarrow \mathsf{PKE.Enc}(\mathsf{sk}, 0)$
	return c

Figure 2.2: Security challenge experiment for KDM security with respect to the family of functions Υ.

Definition 21 (KDM Security)**.** We say SKE is *key dependent message (KDM) secure* (see e.g.[BHHO08], also known as *circular secure*) with respect to the family of functions Υ from the secret key space of SKE to the message space of SKE, if for every PPT adversary $\mathcal{A}$ that only queries $f \in \Upsilon$ the probability

$$\mathrm{Adv}^{\mathrm{kdm}}_{\mathsf{SKE},\Upsilon,\mathcal{A}}(\lambda) := \left| \Pr\left[\mathrm{Exp}^{\mathrm{kdm}}_{\mathsf{SKE},\Upsilon,\mathcal{A}}(\lambda) = 1\right] - \frac{1}{2} \right|$$

is negligible in λ, where $\mathrm{Exp}^{\mathrm{kdm}}_{\mathsf{SKE},\Upsilon,\mathcal{A}}(\lambda)$ is as defined in Figure 2.2.

2.6 Message Authentication Codes and Signatures

Definition 22 (MAC)**.** A *message authentication code (MAC)* is a tuple of PPT algorithms $\mathsf{MAC} := (\mathsf{Gen}, \mathsf{Tag}, \mathsf{Ver})$ such that:

$\mathsf{Gen}(1^\lambda)$: On input of the security parameter λ in unary represenation, generates public parameters pp and a secret key sk.

$\mathsf{Tag}(\mathsf{pp}, \mathsf{sk}, m)$: On input of public parameters pp, the secret key sk and a message $m \in \mathcal{M}$, returns a tag tag.

$\mathsf{Ver}(\mathsf{pp}, \mathsf{sk}, m, \mathsf{tag})$: On input of the public parameters pp, secret key sk , message m and tag tag, outputs a bit $b \in \{0,1\}$ (corresponding to the validity of tag respective to m).

We say MAC is *perfectly correct*, if for all $\lambda \in \mathbb{N}$,all $m \in \mathcal{M}$ and all $(\mathsf{pp}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$ we have

$$\mathsf{Ver}(\mathsf{pp}, \mathsf{sk}, m, \mathsf{Tag}(\mathsf{pp}, \mathsf{sk}, m)) = 1.$$

Definition 23 ($\mathsf{EUF\text{-}CMA}$ security)**.** Let $\mathsf{MAC} := (\mathsf{Gen}, \mathsf{Tag}, \mathsf{Ver})$ be a MAC. For any adversary $\mathcal{A}$, we define the following experiment:

$\underline{\mathrm{Exp}^{\text{euf-cma}}_{\mathsf{MAC},\mathcal{A}}(\lambda)}$:	$\underline{\mathcal{O}_{\mathrm{tag}}(m)}$:
$(\mathsf{pp}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$	$\mathcal{Q}_{\mathrm{tag}} := \mathcal{Q}_{\mathrm{tag}} \cup \{m\}$
$\mathcal{Q}_{\mathrm{tag}} := \emptyset$	$\mathsf{tag} \leftarrow \mathsf{Tag}(\mathsf{pp}, \mathsf{sk}, m)$
$(m^\star, \mathsf{tag}^\star) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathrm{tag}}(\cdot)}(\mathsf{pp})$	**return** tag
if $m^\star \notin \mathcal{Q}_{\mathrm{tag}} \wedge \mathcal{O}_{\mathrm{ver}}(m^\star, \mathsf{tag}^\star) = 1$	
$\quad$ **return** 1	$\underline{\mathcal{O}_{\mathrm{ver}}(m, \mathsf{tag})}$:
else return 0	$b \leftarrow \mathsf{Ver}(\mathsf{pp}, \mathsf{sk}, m, \mathsf{tag})$
	return b

The adversary is restricted to one call to $\mathcal{O}_{\text{ver}}$. We say that a MAC scheme MAC is $\mathsf{EUF\text{-}CMA}$ *secure*, if for all PPT adversaries $\mathcal{A}$,

$$\mathrm{Adv}^{\text{euf-cma}}_{\mathsf{MAC},\mathcal{A}}(\lambda) := \Pr[\mathrm{Exp}^{\text{euf-cma}}_{\mathsf{MAC},\mathcal{A}}(\lambda) = 1] \leq \mathsf{negl}(\lambda).$$

Note that in our notion of $\mathsf{EUF\text{-}CMA}$ security, the adversary gets only one forgery attempt. This is due to the fact that we employ the MAC primarily as a building block for our signature. Our notion suffices for this purpose, as an adversary can check the validity of a signature itself.

Definition 24 (Signature)**.** A *signature scheme* is a tuple of PPT algorithms $\mathsf{SIG} := (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$ such that:

$\mathsf{Gen}(1^\lambda)$: On input of the security parameter λ in unary representation, generates a pair $(\mathsf{pk}, \mathsf{sk})$ of keys.

$\mathsf{Sign}(\mathsf{sk}, m)$: On input of the secret key sk and a message $m \in \mathcal{M}$, returns a signature σ.

$\mathsf{Ver}(\mathsf{pk}, m, \sigma)$: On input of a verification key pk, message m and signature σ, outputs a bit $b \in \{0, 1\}$ (corresponding to the validity of σ respective to m).

We say that SIG is *perfectly correct*, if for all $\lambda \in \mathbb{N}$,all $m \in \mathcal{M}$ and all $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$,

$$\mathsf{Ver}(\mathsf{pk}, m, \mathsf{Sign}(\mathsf{sk}, m)) = 1.$$

In bilinear pairing groups, we say a signature scheme SIG is *structure-preserving* if its public keys, signing messages, signatures contain only group elements and verification proceeds via only a set of pairing product equations.

Note that we always implicitly assume the secret key to contain the public key.

Definition 25 ($\mathsf{EUF\text{-}CMA}$ security)**.** For a signature scheme $\mathsf{SIG} := (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$ and any adversary $\mathcal{A}$, we define the following experiment:

$\underline{\mathrm{Exp}^{\text{euf-cma}}_{\mathsf{SIG},\mathcal{A}}(\lambda)}$:	$\underline{\mathcal{O}_{\text{sign}}(m)}$:
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$	$\mathcal{Q}_{\text{sign}} := \mathcal{Q}_{\text{sign}} \cup \{m\}$
$\mathcal{Q}_{\text{sign}} := \emptyset$	$\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$
$(m^\star, \sigma^\star) \leftarrow \mathcal{A}^{\mathcal{O}_{\text{sign}}(\cdot)}(\mathsf{pk})$	**return** σ
if $m^\star \notin \mathcal{Q}_{\text{sign}} \wedge \mathsf{Ver}(\mathsf{pk}, m^\star, \sigma^\star) = 1$	
return 1	
else return 0	

We say that a signature scheme SIG is $\mathsf{EUF\text{-}CMA}$, if for all PPT adversaries $\mathcal{A}$,

$$\mathrm{Adv}^{\text{euf-cma}}_{\mathsf{SIG},\mathcal{A}}(\lambda) := \Pr[\mathrm{Exp}^{\text{euf-cma}}_{\mathsf{SIG},\mathcal{A}}(\lambda) = 1] \leq \mathsf{negl}(\lambda).$$

2.7 Homomorphic Secret Sharing

We consider homomorphic secret sharing (HSS) as introduced in [BGI16a]. By default, throughout this thesis, the term HSS refers to a public-key variant of HSS. Unlike [BGI16a], we do not need to consider non-negligible δ error failure probability.

Definition 26 (Homomorphic Secret Sharing)**.** A (2-party, public-key) *Homomorphic Secret Sharing (HSS)* scheme for a class of programs $\mathcal{P}$ over a ring R with input space $\mathcal{I} \subseteq R$ consists of PPT algorithms $(\mathsf{HSS.Gen}, \mathsf{HSS.Enc}, \mathsf{HSS.Eval})$ with the following syntax:

- $\mathsf{HSS.Gen}(1^\lambda)$: On input a security parameter 1^λ, the key generation algorithm outputs a public key pk and a pair of evaluation keys $(\mathsf{ek}_0, \mathsf{ek}_1)$.
- $\mathsf{HSS.Enc}(\mathsf{pk}, x)$: Given public key pk and secret input value $x \in \mathcal{I}$, the encryption algorithm outputs a ciphertext ct.
- $\mathsf{HSS.Eval}(b, \mathsf{ek}_b, (\mathsf{ct}^{(1)}, \ldots, \mathsf{ct}^{(\rho)}), P, \beta)$: On input party index $b \in \{0,1\}$, evaluation key ek_b, vector of ρ ciphertexts, a program $P \in \mathcal{P}$ with ρ input values and an integer $\beta \geq 2$, the homomorphic evaluation algorithm outputs $y_b \in R_\beta$, constituting party b's share of an output $y \in R_\beta$.

The algorithms $(\mathsf{HSS.Gen}, \mathsf{HSS.Enc}, \mathsf{HSS.Eval})$ should satisfy the following correctness and security requirements:

Correctness: For all $\lambda \in \mathbb{N}$, for all $x^{(1)}, \ldots, x^{(\rho)} \in \mathcal{I}$, for all programs $P \in \mathcal{P}$ with size $|P| \leq \mathsf{poly}(\lambda)$ and $P(x^{(1)}, \ldots, x^{(\rho)}) \neq \perp$, for integer $\beta \geq 2$, for $(\mathsf{pk}, \mathsf{ek}_0, \mathsf{ek}_1) \leftarrow \mathsf{HSS.Gen}(1^\lambda)$ and for $\mathsf{ct}^{(i)} \leftarrow \mathsf{HSS.Enc}(1^\lambda, \mathsf{pk}, x^{(i)})$ we have

$$\Pr^{\mathrm{cor}}_{\mathsf{HSS},(x^{(i)})_i,P,\beta}(\lambda) := \Pr\left[y_0 + y_1 = P(x^{(1)}, \ldots, x^{(\rho)}) \mod \beta\right] \geq 1 - \lambda^{-\omega(1)},$$

where

$$y_b \leftarrow \mathsf{HSS.Eval}(b, \mathsf{ek}_b, (\mathsf{ct}^{(i)})_i, P, \beta)$$

for $b \in \{0,1\}$ and where the probability is taken over the random coins of $\mathsf{HSS.Gen}$, $\mathsf{HSS.Enc}$ and $\mathsf{HSS.Eval}$.

Security: For all security parameters $\lambda \in \mathbb{N}$, for all PPT adversaries $\mathcal{A}$ that on input 1^λ output a bit $b \in \{0,1\}$ (specifying which encryption key to corrupt), and input values $x_0, x_1 \in \mathcal{I}$, we require the following advantage to be negligible in λ:

$$\mathrm{Adv}^{\mathrm{sec}}_{\mathsf{HSS},\mathcal{A}}(\lambda) := \left| \Pr\left[\mathcal{A}(\mathsf{input}_b) = \beta \;\middle|\; \begin{array}{l} (b, x_0, x_1, \mathsf{state}) \leftarrow \mathcal{A}(1^\lambda), \\ \beta \leftarrow \{0,1\}, \\ (\mathsf{pk}, (\mathsf{ek}_0, \mathsf{ek}_1)) \leftarrow \mathsf{HSS.Gen}(1^\lambda), \\ \mathsf{ct} \leftarrow \mathsf{HSS.Enc}(\mathsf{pk}, x_\beta), \\ \mathsf{input}_b := (\mathsf{state}, \mathsf{pk}, \mathsf{ek}_b, \mathsf{ct}) \end{array} \right] - \frac{1}{2} \right|.$$

Within applications, we additionally consider a secret-key variant of HSS.

Definition 27 (Secret-key HSS)**.** *Secret-key HSS* is a weaker notion of HSS, where the role of the public key pk is replaced by a *secret* key sk and where $\mathsf{HSS.Enc}$ is replaced by the following algorithm.

- $\mathsf{HSS.Share}(\mathsf{sk}, x)$: Given secret key sk and secret input value $x \in \mathcal{I}$, the encryption algorithm outputs a pair of shares $(\mathsf{sh}_0, \mathsf{sh}_1)$.

The correctness and security requirements are as above, but where $\mathsf{HSS.Enc}$ is replaced by $\mathsf{HSS.Share}$ and $\mathsf{ct}^{(i)}$, ct are replaced by $\mathsf{sh}_b^{(i)}$, sh_b, respectively.

Chapter 3

CCA-Secure Public-Key Encryption

In this chapter we present a variation of the encryption scheme of Kurosawa and Desmedt [KD04] whose security can be tightly reduced to the decisional Diffie-Hellman assumption. Building on the works of [GHKW16, Hof17] we achieve a tight security reduction via a partitioning argument. In the center of our construction is a compact and efficient pairing-free designated-verifier proof system for the disjunction of two linear languages, which serves as a proof of well-formedness. We start this chapter with a brief overview.

The Decisional Diffie-Hellman assumption. The cryptographic assumption we build on in this chapter is the *decisional Diffie-Hellman (DDH)* assumption. This can be formulated as hardness of deciding subset membership of a linear language: Let $\mathcal{L} := \{[\mathbf{a}] \cdot r \mid r \in \mathbb{Z}_p\} \subseteq \mathbb{G}^2$ for $[\mathbf{a}] \in \mathbb{G}^2$. Then, the DDH assumption states that it is hard to distinguish an element $[\mathbf{t}] \leftarrow_R \mathcal{L}$ from an element $[\mathbf{t}] \leftarrow_R \mathbb{G}^2$. A property which makes the DDH assumption nicely applicable in the context of tight security is its random-self reducibility: Given a DDH-instance $[\mathbf{t}]$, one can re-randomize this instance by computing $[\mathbf{t}'] := [\mathbf{a}] \cdot \alpha + [\mathbf{t}] \cdot \beta$ for $\alpha, \beta \leftarrow_R \mathbb{Z}_p$. If $[\mathbf{t}] \in \mathcal{L}$, then $[\mathbf{t}']$ is distributed uniformly random in $\mathcal{L}$. Further, if $[\mathbf{t}] \notin \mathcal{L}$, then $[\mathbf{a}], [\mathbf{t}]$ is a basis of $\mathbb{G}^2$ and thus $[\mathbf{t}']$ distributed uniformly random in $\mathbb{G}^2$. Therefore, single-instance DDH tightly implies multi-instance DDH.

The Scheme of Kurosawa and Desmedt. The encryption scheme of Kurosawa and Desmedt [KD04] can be described as follows:

$$\begin{aligned} \mathsf{pars} &= [\mathbf{a}] \in \mathbb{G}^2 \\ \mathsf{sk} &= (\mathbf{k}_0, \mathbf{k}_1) \in \mathbb{Z}_p^{2\times 2} \\ \mathsf{pk} &= ([\mathbf{k}_0^\top \mathbf{a}], [\mathbf{k}_1^\top \mathbf{a}]) \in \mathbb{G}^2 \end{aligned}$$

To encrypt a message M, the sender computes

$$\begin{aligned} &[\mathbf{t}] \leftarrow_R \mathcal{L} \\ &[k] = [\mathbf{k}_0^\top \mathbf{a}] \cdot r + H([\mathbf{t}]) \cdot [\mathbf{k}_1^\top \mathbf{a}] \cdot r \end{aligned}$$

and returns

$$([\mathbf{t}], \mathsf{Enc}_{[k]}(M)),$$

where $H \colon \mathbb{G}^2 \to \mathbb{Z}_p$ is a collision resistant hash function and Enc is a symmetric authenticated encryption scheme. The receiver in knowledge of the secret key can decrypt by computing $[k]$ as

$$[k] = \mathbf{k}_0^\top \cdot [\mathbf{t}] + H([\mathbf{t}]) \cdot \mathbf{k}_1^\top \cdot [\mathbf{t}].$$

Note that computing $[k]$ can be viewed as evaluating a 2-*universal* hash proof system for the language $\mathcal{L}$: Even given the public key and the key for a statement $[\mathbf{t}] \notin L$, the secret keys $\mathbf{k}_0, \mathbf{k}_1$ carry enough entropy for the key $[k]$ to be distributed uniformly at random for statements $[\mathbf{t}'] \notin L$.

The security reduction to the decisional Diffie-Hellman assumption proceeds as follows: First, it *always* uses the secret key to compute $[k]$ for the challenge ciphertext. Now, as the witness r is not necessary anymore to compute the challenge ciphertext, the reduction can switch the statement in the challenge ciphertext to uniformly at random from $\mathbb{G}^2$ (instead of $\mathcal{L}$). By the decisional Diffie-Hellman assumption this change is not noticeable to any bounded adversary.

Next, the 2-universality allows to reject all decryption queries with statement outside $\mathcal{L}$: As the key $[k]$ in such decryption queries looks uniformly random from the point of the adversary, any such queries would with overwhelming probability be rejected by the symmetric authenticated encryption scheme Enc. Finally, the entropy left in $\mathbf{k}_0, \mathbf{k}_1$ can be used to argue that the key $[k]$ in the challenge ciphertext looks uniformly at random from the point of the adversary and therefore perfectly hides the encrypted message.

Towards a Tight Security Reduction. Even though the decisional Diffie-Hellman assumption is re-randomizable, going from single- to multi-ciphertext security introduces a security loss for the scheme of Kurosawa-Desmedt: Given many ciphertexts outside $\mathcal{L}$, the security reduction cannot reject decryption queries outside $\mathcal{L}$ anymore, because the entropy in $\mathbf{k}_0, \mathbf{k}_1$ is limited. To overcome this issue, we follow the strategy of [GHKW16] to gradually randomize $\mathbf{k}_0$, such that finally each ciphertext has a individually randomized secret key. Once the key is fully randomized, we can reject decryption queries with $[\mathbf{t}] \notin \mathcal{L}$ and security of the scheme follows via re-randomizability of the decisional Diffie-Hellman assumption.

To randomize $\mathbf{k}_0$ we proceed as follows: Similar to [GHKW16] in the i-th hybrid we add a random offset to half of the ciphertexts (with statement in some language $\mathcal{L}_0 = \{[\mathbf{a}_0] \cdot r \mid r \in \mathbb{Z}_p\}$) and a random offset to the other half of the ciphertexts (with statement in some language $\mathcal{L}_1 = \{[\mathbf{a}_1] \cdot r \mid r \in \mathbb{Z}_p\}$). By choosing the offset in the left-kernel of $\mathbf{a}_0$ (and $\mathbf{a}_1$), this change does not show up in the challenge ciphertexts itself. The problem are decryption queries: In *ill-formed* decryption queries that do not respect the partitioning, this change would be detectable by an adversary. We solve this problem (similar to [Hof17]) by adding a proof of well-formedness to the ciphertexts, enforcing $[\mathbf{t}] \in \mathcal{L} \cup \mathcal{L}_0 \cup \mathcal{L}_1$.

Compact Pairing-Free Or-Proof. Our proof system for the simpler language $[\mathbf{t}] \in \mathcal{L} \cup \mathcal{L}_0$ can be described as follows. The public key consists of two hash proof system public keys for the linear language $\mathcal{L}$ of the form $[\mathbf{k}_x^\top \mathbf{a}], [\mathbf{k}_y^\top \mathbf{a}]$. The idea is to use the evaluation of the first hash proof system to blind the second evaluation whenever $[\mathbf{t}] \in \mathcal{L}_0$. This can be done by computing a linear combination of two vectors which are *linearly dependent* if and only if $[\mathbf{t}] \in \mathcal{L}_0$. More precisely, a proof will be of the form

$$\pi = [\mathbf{a}_0] \cdot x + [\mathbf{t}] \cdot y,$$

where x and y are derived from the keys $[\mathbf{k}_x^\top \mathbf{t}]$ and $[\mathbf{k}_y^\top \mathbf{t}]$, respectively.

Now, even given many proofs for statements $[\mathbf{t}] \in \mathcal{L}$ and $[\mathbf{t}] \in \mathcal{L}_0$, there is entropy left in the secret key $\mathbf{k}_y^\top$ that only shows up on statements $[\mathbf{t}] \notin \mathcal{L} \cup \mathcal{L}_0$, as this entropy is hidden by x whenever the vectors $[\mathbf{a}_0]$ and $[\mathbf{t}]$ are linearly dependent.

Note that the proof only adds *one* group element to the ciphertext size, as we can add one part of the proof to the key used for encryption. We show that the adversary will not be able to provide a valid ciphertext for statements outside $\mathcal{L}_0 \cup \mathcal{L}_1$.

Roadmap. We start this chapter by formally defining the notion of *qualified proof system* in Section 3.1. We want to note that this slightly weird security notion is due to our instantiation and just what we need for our encryption scheme. We give an instantiation based on DDH (and more generally, k-Lin) which only adds one group element (resp. k^2 group elements) to ciphertexts in Section 3.2. We present our encryption scheme as *key encapsulation mechanism (KEM)*, formally separating the key derivation from the symmetric encryption. We give a definition of key encapsulation mechanisms in 3.3 and present our construction in Section 3.4.

The following is taken verbatim (with minor changes) from our work [GHK17].

3.1 Qualified Proof Systems

We consider a combination of a designated verifier proof system and a hash proof system. Combined proofs consist of a proof Π and a key K, where the key K can be recovered by the verifier with a secret key and the proof Π. The key K can be part of the key in the key encapsulation mechanism presented later and thus will not enlarge the ciphertext size.

Definition 28 (Proof system)**.** Let $\mathcal{L} = \{\mathcal{L}_{\mathsf{pars}}\}$ be a family of languages indexed by the public parameters pars, with $\mathcal{L}_{\mathsf{pars}} \subseteq \mathcal{X}_{\mathsf{pars}}$ and an efficiently computable witness relation $\mathcal{R}$. A *proof system* for $\mathcal{L}$ is a tuple of PPT algorithms $(\mathsf{PGen}, \mathsf{PPrv}, \mathsf{PVer}, \mathsf{PSim})$ such that:

$\mathsf{PGen}(1^\lambda)$: On input of the security parameter λ in unary representation, generates a public key ppk and a secret key psk.

$\mathsf{PPrv}(\mathsf{ppk}, x, w)$: Given a word $x \in \mathcal{L}$ and a witness w with $\mathcal{R}(x, w) = 1$, deterministically outputs a proof Π and a key K.

$\mathsf{PVer}(\mathsf{ppk}, \mathsf{psk}, x, \Pi)$: On input ppk, psk, $x \in \mathcal{X}$ and Π, deterministically outputs a verdict $b \in \{0, 1\}$ and in case $b = 1$ additionally a key K, else $\perp$.

$\mathsf{PSim}(\mathsf{ppk}, \mathsf{psk}, x)$: Given the keys ppk, psk and a word $x \in \mathcal{X}$, deterministically outputs a proof Π and a key K.

Our definition of a qualified proof system is a variant of "benign proof systems" given in [Hof17] tailored to our purposes. Compared to benign proof systems, our proof systems feature an additional "key derivation" stage, and satisfy a weaker soundness requirement (that is of course still sufficient for our purpose). We need to weaken the soundness condition (compared to benign proof systems) in order to prove soundness of our instantiation.

We will consider soundness relative to a language $\mathcal{L}^{\mathrm{snd}} \supseteq \mathcal{L}$. An adversary trying to break soundness has access to an oracle simulating proofs and keys for statements randomly chosen from $\mathcal{L}^{\mathrm{snd}} \setminus \mathcal{L}$ and a verification oracle, which only replies other than $\perp$ if the adversary provides a valid proof and has a high a-priori knowledge of the corresponding key. The adversary wins if it can provide a valid verification query

outside $\mathcal{L}^{\text{snd}}$. The adversary loses immediately if it provides a valid verification query in $\mathcal{L}^{\text{snd}} \setminus \mathcal{L}$. This slightly weird condition is necessitated by our concrete instantiation which we do not know how to prove sound otherwise. We will give more details in the corresponding proof in Section 3.2. The weaker notion of soundness still suffices to prove our KEM secure, because we employ soundness at a point where valid decryption queries in $\mathcal{L}^{\text{snd}} \setminus \mathcal{L}$ end the security experiment anyway.

Definition 29 (Qualified proof system)**.** Let $\mathsf{PS} = (\mathsf{PGen}, \mathsf{PPrv}, \mathsf{PVer}, \mathsf{PSim})$ be a proof system for a family of languages $\mathcal{L} = \{\mathcal{L}_{pars}\}$. Let $\mathcal{L}^{\text{snd}} = \{\mathcal{L}^{\text{snd}}_{pars}\}$ be a family of languages, such that $\mathcal{L}_{\mathsf{pars}} \subseteq \mathcal{L}^{\text{snd}}_{\mathsf{pars}}$. We say that PS is $\mathcal{L}^{\text{snd}}$-*qualified*, if the following properties hold:

Completeness: For all possible public parameters pars, for all words $x \in \mathcal{L}$, and all witnesses w such that $\mathcal{R}(x, w) = 1$, we have

$$\Pr[\mathsf{PVer}(\mathsf{ppk}, \mathsf{psk}, x, \Pi) = (1, K)] = 1,$$

where the probability is taken over $(\mathsf{ppk}, \mathsf{psk}) \leftarrow_R \mathsf{PGen}\ (1^\lambda)$ and $(\Pi, K) := \mathsf{PPrv}(\mathsf{ppk}, x, w)$.

Uniqueness of the proofs: For all possible public parameters pars, all key pairs $(\mathsf{ppk}, \mathsf{psk})$ in the output space of $\mathsf{PGen}\ (1^\lambda)$, and all words $x \in \mathcal{L}$, there exists *at most one* Π such that $\mathsf{PVer}(\mathsf{ppk}, \mathsf{psk}, x, \Pi)$ outputs the verdict 1.

Perfect zero-knowledge: For all public parameters pars, all key pairs $(\mathsf{ppk}, \mathsf{psk})$ in the range of $\mathsf{PGen}(1^\lambda)$, all words $x \in \mathcal{L}$, and all witnesses w with $\mathcal{R}(x, w) = 1$, we have

$$\mathsf{PPrv}(\mathsf{ppk}, x, w) = \mathsf{PSim}(\mathsf{ppk}, \mathsf{psk}, x).$$

Constrained $\mathcal{L}^{\text{snd}}$-soundness: For any stateful PPT adversary $\mathcal{A}$, we consider the following soundness game (where PSim and PVer are implicitly assumed to have access to ppk):

$\underline{\text{Exp}^{\text{csnd}}_{\mathsf{PS},\mathcal{A}}(\lambda)}$:	$\underline{\mathcal{O}_{\text{sim}}}$:	$\underline{\mathcal{O}_{\text{ver}}(x, \Pi, \mathsf{pred})}$:
$(ppk, psk) \leftarrow_R \mathsf{PGen}(1^\lambda)$	$x \leftarrow_R \mathcal{L}^{\text{snd}} \setminus \mathcal{L}$	$(v, K) := \mathsf{PVer}(\mathsf{psk}, x, \Pi)$
$\mathcal{A}^{\mathcal{O}_{\text{sim}}, \mathcal{O}_{\text{ver}}(\cdot,\cdot,\cdot)}(1^\lambda, ppk)$	$(\Pi, K) \leftarrow \mathsf{PSim}(\mathsf{psk}, x)$	**if** $v = 1$ **and** $\mathsf{pred}(K) = 1$
if $\mathcal{O}_{\text{ver}}$ **returned** *lose*	**return** (x, Π, K)	**if** $x \in \mathcal{L}$
return 0		**return** K
if $\mathcal{O}_{\text{ver}}$ **returned** *win*		**else if** $x \in \mathcal{L}^{\text{snd}}$
return 1		**return** *lose* **and abort**
return 0		**else return** *win* **and abort**
		else return $\perp$

Let Q_{ver} be the total number of oracle queries to $\mathcal{O}_{\text{ver}}$ and pred_i be the predicate submitted by $\mathcal{A}$ on the i-th query. The adversary $\mathcal{A}$ loses and the experiment aborts if the verification oracle answers *lose* on some query of $\mathcal{A}$. The adversary $\mathcal{A}$ wins, if the oracle $\mathcal{O}_{\text{ver}}$ returns *win* on some query (x, Π, pred) of $\mathcal{A}$ with $x \notin \mathcal{L}^{\text{snd}}$ and the following conditions hold:

- The predicate corresponding to the i-th query is of the form $\mathsf{pred}_i \colon \mathcal{K} \cup \{\bot\} \to \{0,1\}$ with $\mathsf{pred}_i(\bot) = 0$ for all $i \in \{1, \ldots, Q_{\mathrm{ver}}\}$.
- For all environments $\mathcal{E}$ having at most running time of the described constrained soundness experiment, we require that

$$\mathrm{uncert}^{\mathrm{snd}}_{\mathcal{A}}(\lambda) := \frac{1}{Q_{\mathrm{ver}}} \sum_{i=1}^{Q_{\mathrm{ver}}} \Pr_{K \in \mathcal{K}}[\mathsf{pred}_i(K) = 1 \text{ when } \mathcal{A} \text{ runs in } \mathcal{E}]$$

  is negligible in λ.

Note that in particular the adversary cannot win anymore after the verification oracle replied *lose* on one of its queries, as in this case the experiment directly aborts and outputs 0. Let $\mathrm{Adv}^{\mathrm{snd}}_{\mathcal{L}^{\mathrm{snd}},\mathsf{PS},\mathcal{A}}(\lambda) := \Pr[\mathrm{Exp}^{\mathrm{csnd}}_{\mathsf{PS},\mathcal{A}}(\lambda) = 1]$, where the probability is taken over the random coins of $\mathcal{A}$ and $\mathrm{Exp}^{\mathrm{csnd}}_{\mathsf{PS},\mathcal{A}}$. Then we say *constrained* $\mathcal{L}^{\mathrm{snd}}$*-soundness* holds for PS, if for every PPT adversary $\mathcal{A}$, $\mathrm{Adv}^{\mathrm{snd}}_{\mathcal{L}^{\mathrm{snd}},\mathsf{PS},\mathcal{A}}(\lambda) = \mathsf{negl}(\lambda)$.

To prove security of the key encapsulation mechanism later, we need to switch between two proof systems. Intuitively this provides an additional degree of freedom, allowing to randomize the keys of the challenge ciphertexts gradually. To justify this transition, we introduce the following notion of indistinguishable proof systems.

Definition 30 ($\mathcal{L}^{\mathrm{snd}}$-indistinguishability of two proof systems)**.** Let $\mathcal{L} \subseteq \mathcal{L}^{\mathrm{snd}}$ be (families of) languages. Let $\mathsf{PS}_0 := (\mathsf{PGen}_0, \mathsf{PPrv}_0, \mathsf{PVer}_0, \mathsf{PSim}_0)$ and $\mathsf{PS}_1 := (\mathsf{PGen}_1, \mathsf{PPrv}_1, \mathsf{PVer}_1, \mathsf{PSim}_1)$ proof systems for $\mathcal{L}$. For every adversary $\mathcal{A}$, we define the following experiment (where PSim_b and PVer_b are implicitly assumed to have access to ppk):

$\underline{\mathrm{Exp}^{\mathrm{PS-ind}}_{\mathcal{L}^{\mathrm{snd}},\mathsf{PS}_0,\mathsf{PS}_1,\mathcal{A}}(\lambda)}$:	$\underline{\mathcal{O}^b_{\mathrm{sim}}}$:	$\underline{\mathcal{O}^b_{\mathrm{ver}}(x, \Pi, \mathsf{pred})}$:
$b \leftarrow_R \{0,1\}$	$x \leftarrow_R \mathcal{L}^{\mathrm{snd}} \backslash \mathcal{L}$	$(v, K) := \mathsf{PVer}_b(\mathsf{psk}, x, \Pi)$
$(\mathsf{ppk}, \mathsf{psk}) \leftarrow \mathsf{PGen}_b(1^\lambda)$	$(\Pi, K) \leftarrow \mathsf{PSim}_b(\mathsf{psk}, x)$	**if** $v = 1$ **and** $\mathsf{pred}(K) = 1$
$b' \leftarrow \mathcal{A}^{\mathcal{O}^b_{\mathrm{sim}}, \mathcal{O}^b_{\mathrm{ver}}(\cdot,\cdot)}(\mathsf{ppk})$	**return** (x, Π, K)	**and** $x \in \mathcal{L}^{\mathrm{snd}}$
if $b = b'$ **return** 1		**return** K
else return 0		**else return** $\bot$

As soon as $\mathcal{A}$ has submitted one query which is replied with *lose* by the verification oracle, the experiment aborts and outputs 0.

We define the advantage function

$$\mathrm{Adv}^{\text{PS-ind}}_{\mathcal{L}^{\mathrm{snd}},\mathsf{PS}_0,\mathsf{PS}_1,\mathcal{A}}(\lambda) := \left| \Pr\left[\mathrm{Exp}^{\mathrm{PS-ind}}_{\mathcal{L}^{\mathrm{snd}},\mathsf{PS}_0,\mathsf{PS}_1,\mathcal{A}}(\lambda) = 1\right] - \frac{1}{2} \right|.$$

We say PS_0 and PS_1 are $\mathcal{L}^{\mathrm{snd}}$*-indistinguishable*, if for all (unbounded) algorithms $\mathcal{A}$ the advantage $\mathrm{Adv}^{\text{PS-ind}}_{\mathcal{L},\mathsf{PS}_0,\mathsf{PS}_1,\mathcal{A}}(\lambda)$ is negligible in λ.

Note that we adopt a different (and simpler) definition for the verification oracle in the indistinguishability game than in the soundness game, in particular it leaks more information about the keys. We can afford this additional leakage for indistinguishability, but not for soundness.

In order to prove security of the key encapsulation mechanism presented in Section 3.4, we will require one proof system and the existence of a second proof system it can be extended to. We capture this property in the following definition.

Definition 31 ($\mathcal{L}^{\text{ext}}$-extensibility of a proof system)**.** Let $\mathcal{L} \subseteq \mathcal{L}^{\text{snd}} \subseteq \mathcal{L}^{\text{ext}}$ be three (families of) languages. An $\mathcal{L}^{\text{snd}}$-qualified proof system PS for language $\mathcal{L}$ is said to be $\mathcal{L}^{\text{ext}}$-*extensible* if there exists a proof system PS' for $\mathcal{L}$ that complies with $\mathcal{L}^{\text{ext}}$-constrained soundness and such that PS and PS' are $\mathcal{L}^{\text{snd}}$-indistinguishable.

3.2 A Qualified Proof System for Or-Languages

In the following sections we explain how the public parameters $\mathsf{pars}_{\mathsf{PS}}$ are sampled, how our system of or-languages is defined and how to construct a qualified proof system complying with constrained soundness respective to these languages.

First, we need to choose a $k \in \mathbb{N}$ depending on the assumption we use to prove security of our constructions. We invoke $\mathsf{GGen}(1^\lambda)$ to obtain a group description $\mathcal{G} = (\mathbb{G}, p, P)$ with $|\mathbb{G}| \geq 2^{2\lambda}$. Next, we sample matrices $\mathbf{A} \leftarrow_R \mathcal{D}_{2k,k}$ and $\mathbf{A}_0 \leftarrow_R \mathcal{U}_{2k,k}$, where we assume without loss of generality that $\overline{\mathbf{A}}_0$ is full rank. Let $\mathcal{H}_0$ and $\mathcal{H}_1$ be *universal* hash function generators returning functions of the form $\mathsf{h}_0 \colon \mathbb{G}^{k^2+1} \to \mathbb{Z}_p^{k\times k}$ and $\mathsf{h}_1 \colon \mathbb{G}^{k+1} \to \mathbb{Z}_p^k$ respectively. Let $\mathsf{h}_0 \leftarrow_R \mathcal{H}_0$ and $\mathsf{h}_1 \leftarrow_R \mathcal{H}_1$.

Altogether, we define the public parameters for our proof system to comprise

$$\mathsf{pars}_{\mathsf{PS}} := (k, \mathcal{G}, [\mathbf{A}], [\mathbf{A}_0], \mathsf{h}_0, \mathsf{h}_1).$$

We assume from now that all algorithms have access to $\mathsf{pars}_{\mathsf{PS}}$ without explicitly stating it as input.

Additionally, let $\mathbf{A}_1 \in \mathbb{Z}_p^{2k\times k}$ be a matrix distributed according to $\mathcal{U}_{2k,k}$ with the restriction $\overline{\mathbf{A}}_0 = \overline{\mathbf{A}}_1$. Then, we define the languages

$$\begin{aligned}
\mathcal{L} &:= \operatorname{span}([\mathbf{A}]),\\
\mathcal{L}^{\text{snd}} &:= \operatorname{span}([\mathbf{A}]) \cup \operatorname{span}([\mathbf{A}_0]),\\
\mathcal{L}^{\text{ext}} &:= \operatorname{span}([\mathbf{A}]) \cup \operatorname{span}([\mathbf{A}_0]) \cup \operatorname{span}([\mathbf{A}_1]).
\end{aligned}$$

A crucial building block for the key encapsulation mechanism will be a proof system PS that is $\mathcal{L}^{\text{snd}}$-qualified and $\mathcal{L}^{\text{ext}}$-extensible. We give a construction based on $\mathcal{D}_{k^2+1,k}$-MDDH in the following section.

Our goal is to construct an $\mathcal{L}^{\text{snd}}$-qualified proof system for $\mathcal{L}$ based on $\mathcal{D}_{k^2+1,k}$-MDDH for any matrix distribution $\mathcal{D}_{k^2+1,k}$ (see Theorem 1). We give the proof system $\mathsf{PS} := (\mathsf{PGen}, \mathsf{PPrv}, \mathsf{PVer}, \mathsf{PSim})$ for $\mathcal{L}$ in Fig. 3.1.We prove in Theorem 32 that PS indeed meets our requirements. In case $k = 1$ a combined proof requires a single group element for the proof plus an additional group element for the key. Intuitively, our combined proofs consist of a hash proof system respective to the language $\mathcal{L}$ which is protected by a special kind of encryption (depending on the statement itself) to preserve soundness even in the presence of many simulated proofs for statements in $\mathcal{L}^{\text{snd}}$. Note that it suffices to compute merely the diagonal of $[\mathbf{K}]$ instead of the full matrix. With the current presentation we aim to improve readability.

$\underline{\mathsf{PGen}(1^\lambda)}$:
$\mathbf{K_X} \leftarrow_R \mathbb{Z}_p^{(k^2+1)\times 2k}$
$\mathbf{K_y} \leftarrow_R \mathbb{Z}_p^{(k+1)\times 2k}$
$\mathsf{ppk} := ([\mathbf{K_X A}], [\mathbf{K_y A}])$
$\mathsf{psk} := (\mathbf{K_X}, \mathbf{K_y})$
return $(\mathsf{ppk}, \mathsf{psk})$

$\underline{\mathsf{PVer}(\mathsf{ppk}, \mathsf{psk}, [\mathbf{t}], [\pi^\star])}$:
$\overline{\mathbf{X}} := \mathsf{h}_0(\mathbf{K_X}[\mathbf{t}]) \in \mathbb{Z}_p^{k\times k}$
$\mathbf{y} := \mathsf{h}_1(\mathbf{K_y}[\mathbf{t}]) \in \mathbb{Z}_p^{k}$
$[\pi] := \overline{[\mathbf{A}_0]} \cdot \mathbf{X} + \overline{[\mathbf{t}]} \cdot \mathbf{y}^\top \in \mathbb{G}^{k\times k}$
$[\mathbf{K}] := \underline{[\mathbf{A}_0]} \cdot \mathbf{X} + \underline{[\mathbf{t}]} \cdot \mathbf{y}^\top \in \mathbb{G}^{k\times k}$
$[\kappa] := \underline{\mathrm{trace}}([\mathbf{K}]) \in \mathbb{G}$
if $[\pi] = [\pi^\star]$ **return** $(1, [\kappa])$
else return $(0, \bot)$

$\underline{\mathsf{PPrv}(\mathsf{ppk}, [\mathbf{t}], \mathbf{r})}$:
$\overline{\mathbf{X}} := \mathsf{h}_0([\mathbf{K_X A}]\mathbf{r}) \in \mathbb{Z}_p^{k\times k}$
$\mathbf{y} := \mathsf{h}_1([\mathbf{K_y A}]\mathbf{r}) \in \mathbb{Z}_p^{k}$
$[\pi] := \overline{[\mathbf{A}_0]} \cdot \mathbf{X} + \overline{[\mathbf{t}]} \cdot \mathbf{y}^\top \in \mathbb{G}^{k\times k}$
$[\mathbf{K}] := \underline{[\mathbf{A}_0]} \cdot \mathbf{X} + \underline{[\mathbf{t}]} \cdot \mathbf{y}^\top \in \mathbb{G}^{k\times k}$
$[\kappa] := \underline{\mathrm{trace}}([\mathbf{K}]) \in \mathbb{G}$
return $([\pi], [\kappa])$

$\underline{\mathsf{PSim}(\mathsf{ppk}, \mathsf{psk}, [\mathbf{t}])}$:
$\overline{\mathbf{X}} := \mathsf{h}_0(\mathbf{K_X}[\mathbf{t}]) \in \mathbb{Z}_p^{k\times k}$
$\mathbf{y} := \mathsf{h}_1(\mathbf{K_y}[\mathbf{t}]) \in \mathbb{Z}_p^{k}$
$[\pi] := \overline{[\mathbf{A}_0]} \cdot \mathbf{X} + \overline{[\mathbf{t}]} \cdot \mathbf{y}^\top \in \mathbb{G}^{k\times k}$
$[\mathbf{K}] := \underline{[\mathbf{A}_0]} \cdot \mathbf{X} + \underline{[\mathbf{t}]} \cdot \mathbf{y}^\top \in \mathbb{G}^{k\times k}$
$[\kappa] := \underline{\mathrm{trace}}([\mathbf{K}]) \in \mathbb{G}$
return $([\pi], [\kappa])$

Figure 3.1: $\mathcal{L}^{\mathrm{snd}}$-qualified proof system PS for $\mathcal{L}$.

Theorem 32. *If the $\mathcal{D}_{k^2+1,k}$-MDDH assumption holds in $\mathbb{G}$, and h_0, h_1 are universal hash functions, then the proof system PS described in Fig. 3.1 is $\mathcal{L}^{\mathrm{snd}}$-qualified. Further, the proof system PS is $\mathcal{L}^{\mathrm{ext}}$-extensible.*

Proof. Completeness and *perfect zero-knowledge* follow straightforwardly from the fact that for all $\mathbf{t} = [\mathbf{A}]\mathbf{r}$ for an $\mathbf{r} \in \mathbb{Z}_p^k$ it holds $[\mathbf{K_X A}]\mathbf{r} = \mathbf{K_X}[\mathbf{t}]$ and $[\mathbf{K_y A}]\mathbf{r} = \mathbf{K_y}[\mathbf{t}]$.

Uniqueness of the proofs follows from the fact that the verification algorithm computes a unique proof $[\pi]$ and aborts if $[\pi] \neq [\pi^\star]$.

We prove in Theorem 33 that PS satisfies constrained $\mathcal{L}^{\mathrm{snd}}$-soundness.

For $\mathcal{L}^{\mathrm{ext}}$-extensibility we refer to Section 3.2. In Fig. 3.4, we describe a proof system PS' for $\mathcal{L}$, in Theorem 34 we prove that PS and PS' are $\mathcal{L}^{\mathrm{snd}}$-indistinguishable, and in Theorem 35 that PS' complies with constrained $\mathcal{L}^{\mathrm{ext}}$-soundness. □ □

Lemma 33 (Constrained $\mathcal{L}^{\mathrm{snd}}$-soundness of PS)**.** *If the $\mathcal{D}_{k^2+1,k}$-MDDH assumption holds in $\mathbb{G}$ and h_0 and h_1 are universal hash functions, then the proof system described in Fig. 3.1 complies with constrained $\mathcal{L}^{\mathrm{snd}}$-soundness. Namely, for any adversary $\mathcal{A}$ against $\mathcal{L}^{\mathrm{snd}}$-soundness, there exists an adversary $\mathcal{B}$ such that $T(\mathcal{B}) \approx T(\mathcal{A}) + (Q_{\mathrm{sim}} + Q_{\mathrm{ver}}) \cdot \mathsf{poly}(\lambda)$ and*

$$\begin{aligned}\mathrm{Adv}^{\mathrm{snd}}_{\mathcal{L}^{\mathrm{snd}},\mathsf{PS},\mathcal{A}}(\lambda) \leq \mathrm{Adv}^{\mathrm{mddh}}_{\mathbb{G},\mathcal{B},\mathcal{D}_{k^2+1,k}}(\lambda) + Q_{\mathrm{ver}} \cdot \mathrm{uncert}^{\mathrm{snd}}_{\mathcal{A}}(\lambda) \\ + (Q_{\mathrm{sim}} + Q_{\mathrm{ver}} + 1) \cdot 2^{-\Omega(\lambda)},\end{aligned}$$

where Q_{sim}, Q_{ver} are the number of calls to $\mathcal{O}_{\mathrm{sim}}$ and $\mathcal{O}_{\mathrm{ver}}$ respectively, $\mathrm{uncert}^{\mathrm{snd}}_{\mathcal{A}}(\lambda)$ *describes the uncertainty of the predicates provided by $\mathcal{A}$ and* poly *is a polynomial function, independent of $T(\mathcal{A})$.*

#	**sim. X for** $[\mathbf{t}] \in \mathcal{L}^{\mathrm{snd}} \backslash \mathcal{L}$	**ver.** $[\mathbf{K}]$ **for** $[\mathbf{t}] \notin \mathcal{L}$	**game knows**	**remark**
$\mathbf{G}_0$	$\mathbf{X} := \mathsf{h}_0(\mathbf{K}_{\mathbf{X}}[\mathbf{t}])$	$\underline{[\mathbf{A}_0]} \cdot \mathbf{X} + \underline{[\mathbf{t}]} \cdot \mathbf{y}^\top$		$\mathcal{L}^{\mathrm{snd}}$-soundn. game w/o *lose*
$\mathbf{G}_1$	$\mathbf{X} := \mathsf{h}_0(\mathbf{K}_{\mathbf{X}}[\mathbf{t}])$	$\underline{\mathbf{A}_0}\overline{\mathbf{A}_0}^{-1}\left([\pi^\star] - \overline{[\mathbf{t}]} \cdot \mathbf{y}^\top\right) + \underline{[\mathbf{t}]} \cdot \mathbf{y}^\top$	$\mathbf{A}, \mathbf{A}_0$	win. chances increase
$\mathbf{G}_2$	$\mathbf{u} \leftarrow_R \mathbb{Z}_p^{k^2+1}$, $\mathbf{X} := \mathsf{h}_0([\mathbf{u}])$	$\underline{\mathbf{A}_0}\overline{\mathbf{A}_0}^{-1}\left([\pi^\star] - \overline{[\mathbf{t}]} \cdot \mathbf{y}^\top\right) + \underline{[\mathbf{t}]} \cdot \mathbf{y}^\top$	$\mathbf{A}, \mathbf{A}_0$	$\mathcal{D}_{k^2+1,k}$-MDDH
$\mathbf{G}_3$	$\mathbf{X} \leftarrow_R \mathbb{Z}_p^{k \times k}$	$\underline{\mathbf{A}_0}\overline{\mathbf{A}_0}^{-1}\left([\pi^\star] - \overline{[\mathbf{t}]} \cdot \mathbf{y}^\top\right) + \underline{[\mathbf{t}]} \cdot \mathbf{y}^\top$	$\mathbf{A}, \mathbf{A}_0$	Theorem 15 (LOHL)

Figure 3.2: Overview of the proof of $\mathcal{L}^{\mathrm{snd}}$-constrained soundness of PS. The first column shows how $\mathbf{X}$ is computed for queries to $\mathcal{O}_{\mathrm{sim}}$. The second column shows how the pre-key $[\mathbf{K}]$ is computed by the verifier in queries to $\mathcal{O}_{\mathrm{ver}}$ for $[\mathbf{t}] \notin \mathcal{L}$. Recall that the key $[\kappa]$ is the trace of $[\mathbf{K}]$.

Note that, as explained in Section 3.3, in the proof of IND-CCA security of the final hybrid encryption scheme (where we will employ constrained $\mathcal{L}^{\mathrm{snd}}$-soundness of PS to prove IND-CCCA security of our KEM), the term $\mathrm{uncert}^{\mathrm{snd}}_{\mathcal{A}}(\lambda)$ will be statistically small, so we can afford to get a security loss of $Q_{\mathrm{ver}} \cdot \mathrm{uncert}^{\mathrm{snd}}_{\mathcal{A}}(\lambda)$ without compromising tightness.

Proof. We prove $\mathcal{L}^{\mathrm{snd}}$-*soundness* of PS via a series of games, described in Fig. 3.2. We start by giving a short overview of the proof. One can view $[\pi, \mathbf{K}]$ as a special kind of encryption of $\mathbf{y}$. The idea is to first randomize $\mathbf{X}$ used in simulated proofs of statements $[\mathbf{t}] \in \mathcal{L}^{\mathrm{snd}} \setminus \mathcal{L}$, using the $\mathcal{D}_{k^2+1,k}$-MDDH assumption and the Leftover Hash Lemma (Theorem 15). This will make the encryption $[\pi, \mathbf{K}]$ lossy if and only if $[\mathbf{t}] \in \mathrm{span}([\mathbf{A}_0])$. Namely, for $[\mathbf{t}] = [\mathbf{A}_0\mathbf{r}]$ we have $[\pi, \mathbf{K}] = [\mathbf{A}_0] \cdot \mathbf{X} + [\mathbf{t}] \cdot \mathbf{y}^\top = [\mathbf{A}_0](\mathbf{X} + \mathbf{r} \cdot \mathbf{y}^\top)$ and thus $\mathbf{y}$, and in particular $\mathbf{K}_{\mathbf{y}}$, are completely hidden by the randomized $\mathbf{X}$.

In the final proof step we can thus argue that simulation queries leak no information about $\mathbf{K}_{\mathbf{y}}$ apart from what is already contained in the public key and therefore an adversary cannot do better than guessing $[\kappa]$ for a statement outside $\mathcal{L}^{\mathrm{snd}}$.

We start with the constrained $\mathcal{L}^{\mathrm{snd}}$-soundness game, which we refer to as game $\mathbf{G}$. In the following we want to bound the probability

$$\varepsilon := \mathrm{Adv}^{\mathrm{snd}}_{\mathsf{PS},\mathcal{A}}(\lambda).$$

We denote the probability that the adversary $\mathcal{A}$ wins the game $\mathbf{G}_i$ by

$$\varepsilon_i := \mathrm{Adv}_{\mathbf{G}_i,\mathcal{A}}(\lambda).$$

Transition $\mathbf{G} \rightsquigarrow \mathbf{G}_0$: From game $\mathbf{G}_0$ on, on a valid verification query $([\mathbf{t}], \Pi, \mathsf{pred})$ the verification oracle will not return *lose* and abort anymore, but instead simply return $\bot$. This can only increase the winning chances of an adversary $\mathcal{A}$. Thus we obtain

$$\varepsilon \leq \varepsilon_0.$$

Transition $\mathbf{G}_0 \rightsquigarrow \mathbf{G}_1$: We show that $\varepsilon_1 \geq \varepsilon_0$. The difference between $\mathbf{G}_0$ and $\mathbf{G}_1$ is that from game $\mathbf{G}_1$ on the oracle $\mathcal{O}_{\text{ver}}$, on input $([\mathbf{t}], \Pi, \mathsf{pred})$, first checks if $[\mathbf{t}] \in \text{span}([\mathbf{A}])$. If this is the case, $\mathcal{O}_{\text{ver}}$ behaves as in game $\mathbf{G}_0$. Otherwise, it does not check if $[\pi^\star] = [\pi]$ anymore, and it computes

$$[\mathbf{K}] = \underline{\mathbf{A}_0}\overline{\mathbf{A}_0}^{-1}\left([\pi^\star] - \overline{[\mathbf{t}]} \cdot \mathbf{y}^\top\right) + \underline{[\mathbf{t}]} \cdot \mathbf{y}^\top,$$

where $\mathbf{y}$ is computed as in $\mathbf{G}_0$. Note that this computation requires to know $\mathbf{A}_0$, but not $\mathbf{K}_\mathbf{X}$, since $\mathbf{X}$ is not computed explicitly. This will be crucial for the transition to game $\mathbf{G}_2$.

Again we have to show that this can only increase the winning chances of the adversary. In particular, we have to show that this change does not affect the adversaries view on non-winning queries.

First, from game $\mathbf{G}_0$ on the verification oracle $\mathcal{O}_{\text{ver}}$ always returns $\perp$ on queries from $\mathcal{L}^{\text{snd}}\backslash\mathcal{L}$, and thus games $\mathbf{G}_0$ and $\mathbf{G}_1$ only differ when $\mathcal{O}_{\text{ver}}$ is queried on statements with $[\mathbf{t}] \notin \mathcal{L}^{\text{snd}}$. Therefore, it remains to show that for any query $([\mathbf{t}], [\pi^\star], \mathsf{pred})$ to $\mathcal{O}_{\text{ver}}$ with $[\mathbf{t}] \notin \mathcal{L}^{\text{snd}}$, we have that if the query is winning in $\mathbf{G}_0$, then it is also winning in $\mathbf{G}_1$. Suppose $([\mathbf{t}], [\pi^\star], \text{pred})$ satisfies the winning condition in $\mathbf{G}_0$. Then, it must hold true that $[\pi^\star] = \overline{[\mathbf{A}_0]} \cdot \mathbf{X} + \overline{[\mathbf{t}]} \cdot \mathbf{y}^\top$ and $\mathsf{pred}\,(\text{trace}([\mathbf{K}^\star])) = 1$ for $[\mathbf{K}^\star] = \underline{[\mathbf{A}_0]} \cdot \mathbf{X} + \underline{[\mathbf{t}]} \cdot \mathbf{y}^\top$.

In $\mathbf{G}_1$, the pre-key is computed as

$$\underline{\mathbf{A}_0}\overline{\mathbf{A}_0}^{-1}\left([\pi^\star] - \overline{[\mathbf{t}]} \cdot \mathbf{y}^\top\right) + \underline{[\mathbf{t}]} \cdot \mathbf{y}^\top = \underline{[\mathbf{A}_0]} \cdot \mathbf{X} + \underline{[\mathbf{t}]} \cdot \mathbf{y}^\top = [\mathbf{K}^\star],$$

and thus the query is also winning in $\mathbf{G}_1$.

Note that for this step it is crucial that we only require a weakened soundness condition of our proof systems (compared to benign proof systems [Hof17]). Namely, if instead the verification oracle in the soundness experiment $\mathcal{O}_{\text{ver}}$ returned the key $[\kappa]$ for valid statements $[\mathbf{t}] \in \mathcal{L}^{\text{snd}}\backslash\mathcal{L}$, we could not argue that the proof transition does necessarily at most increase the winning chances of an adversary. This holds true as in game $\mathbf{G}_1$ on a statement $[\mathbf{t}] \in \mathcal{L}^{\text{snd}}\backslash\mathcal{L}$ with non-valid proof (but with valid predicate respective to the proof) the key would be returned, whereas in game $\mathbf{G}_0$ "$\perp$" would be returned.

Transition $\mathbf{G}_1 \rightsquigarrow \mathbf{G}_2$: In this transition, we use the $\mathcal{D}_{k^2+1,k}$-MDDH assumption to change the way $\mathbf{X}$ is computed in simulated proofs. More precisely, we will proceed in intermediary games $\mathbf{G}_{1.1}$, $\mathbf{G}_{1.2}$ and $\mathbf{G}_{1.3}$ (see Figure 3.3).

First, as $\mathbf{K}_\mathbf{X}$ and $\mathbf{K}'_\mathbf{X} + \mathbf{U}(\mathbf{A}^\perp)^\top$ for $\mathbf{U} \leftarrow_R \mathbb{Z}_p^{(k^2+1)\times k}$ and $\mathbf{A}^\perp \in \text{orth}(\mathbf{A})$ are distributed equally, we have

$$\varepsilon_{1.1} = \varepsilon_1.$$

Second, with overwhelming probablity over the choices of $\mathbf{A}$, $\mathbf{A}_0$, the matrix $(\mathbf{A}^\perp)^\top\mathbf{A}_0 \in \mathbb{Z}_p^{k\times k}$ is invertible, which implies that $(\mathbf{K}_\mathbf{X} + \mathbf{U}(\mathbf{A}^\perp)^\top)\mathbf{A}_0$ is distributed uniformly random over $\mathbb{Z}_p^{(k^2+1)\times k}$ (even under knowledge of $[\mathbf{K}_\mathbf{X}\mathbf{A}]$). Thus, switching between $(\mathbf{K}_\mathbf{X} + \mathbf{U}(\mathbf{A}^\perp)^\top)\mathbf{A}_0$ and $\mathbf{W}\mathbf{B}(\overline{\mathbf{B}}^k)^{-1}$ is statistically indistinguishable to $\mathcal{A}$ (where $\overline{\mathbf{B}}^k \in \mathbb{Z}_p^{k\times k}$ denotes the upper square matrix of

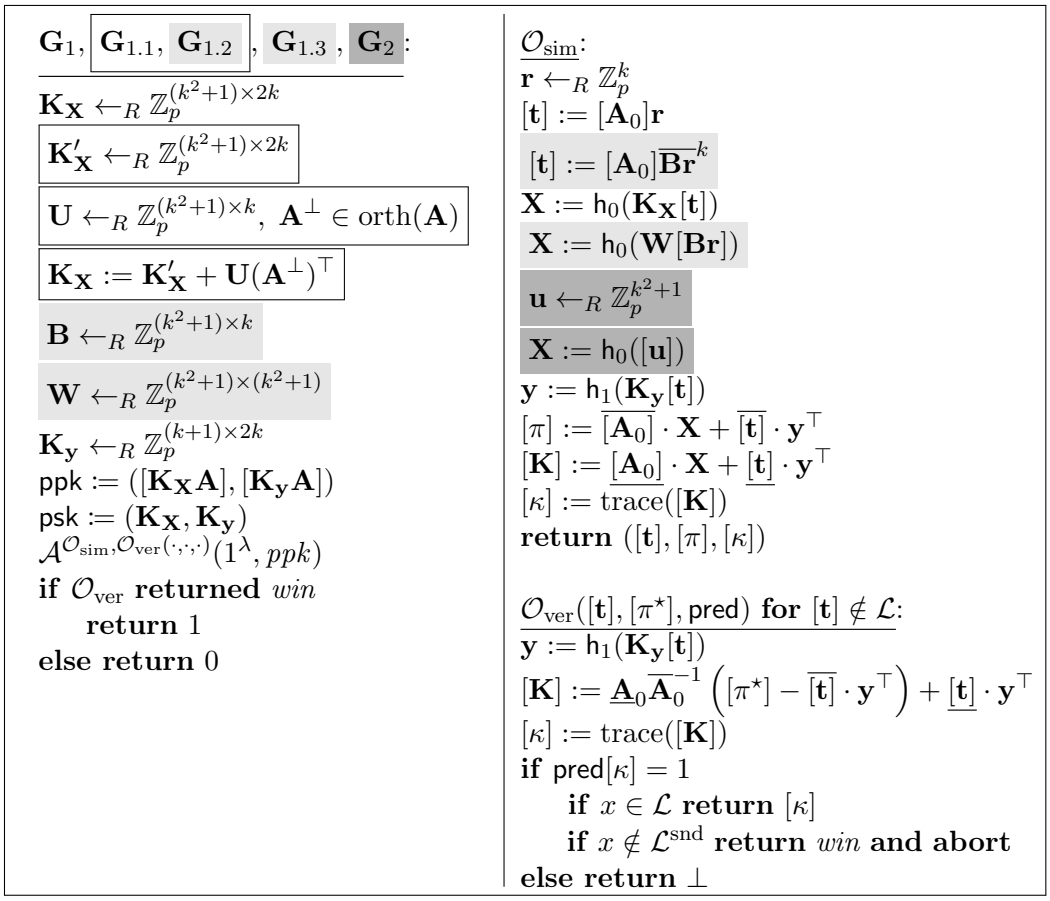

Figure 3.3: Games $\mathbf{G}_1, \mathbf{G}_{1.1}, \mathbf{G}_{1.2}, \mathbf{G}_{1.3}, \mathbf{G}_2$ in the proof of Lemma 33, where the verification oracle for $[\mathbf{t}] \in \mathcal{L}$ is omitted. For $\mathbf{Br} \in \mathbb{Z}_p^{k^2+1}$ by $\overline{\mathbf{Br}}^k \in \mathbb{Z}_p^k$ we denote the vector comprising the upper k entries.

$\mathbf{B}$). Further, we have that $\mathbf{r} \leftarrow_R \mathbb{Z}_p^k$ is distributed equally to $\overline{\mathbf{Br}}^k$ for $\mathbf{r} \leftarrow \mathbb{Z}_p^k$, $\mathbf{B} \leftarrow_R Z_p^{(k^2+1)\times k}$ and

$$\begin{aligned}\mathbf{X} = \mathsf{h}_0(\mathbf{K_X}[\mathbf{t}_i]) = \mathsf{h}_0((\mathbf{K'_X} + \mathbf{U}(\mathbf{A}^\perp)^\top)[\mathbf{A}_0\overline{\mathbf{Br}}^k]) \equiv_s \mathsf{h}_0(\mathbf{WB}(\overline{\mathbf{B}}^k)^{-1}[\overline{\mathbf{Br}}^k]) \\ = \mathsf{h}_0(\mathbf{W}[\mathbf{Br}]).\end{aligned}$$

This yields

$$|\varepsilon_{1.2} - \varepsilon_{1.1}| \leq 2^{-\Omega(\lambda)}.$$

Next, we reverse transition $\mathbf{G}_1 \rightsquigarrow \mathbf{G}_{1.1}$. Note that this change does not affect the public key or any verification query (as from game $\mathbf{G}_1$ on for verification queries outside $\mathrm{span}(\mathbf{A})$ the key $\mathbf{K_X}$ is not employed anymore) or any simulation query (as from game $\mathbf{G}_{1.2}$ on the key $\mathbf{K_X}$ is not employed anymore) and we thus obtain

$$\varepsilon_{1.3} = \varepsilon_{1.2}.$$

Now, let $([\mathbf{B}], [\mathbf{h}_1, \ldots, \mathbf{h}_{Q_{\mathrm{sim}}}])$ be a Q_{sim}-fold $\mathcal{U}_{k^2+1,k}$-MDDH challenge. First, $\mathcal{B}$ picks $\mathbf{A}$ and $\mathbf{A}_0$ as described in Section 3.2, draws $\mathbf{K_X} \leftarrow_R \mathbb{Z}_p^{(k^2+1)\times 2k}$ and

$\mathbf{K_y} \leftarrow_R \mathbb{Z}_p^{(k+1)\times 2k}$ and sends $[\mathbf{A}]$, $[\mathbf{A}_0]$ and $\mathsf{ppk} := ([\mathbf{K_X A}], [\mathbf{K_y A}])$ to $\mathcal{A}$. Further, $\mathcal{B}$ chooses a matrix $\mathbf{W} \leftarrow_R \mathbb{Z}_p^{(k^2+1)\times(k^2+1)}$.

Verification queries are answered by $\mathcal{B}$ according to $\mathcal{O}_{\text{ver}}$.

On the i-th query to $\mathcal{O}_{\text{sim}}$, for all $i \in \{1, \ldots, Q_{\text{sim}}\}$, the adversary $\mathcal{B}$ defines $[\mathbf{t}_i] := \mathbf{A}_0 \overline{[\mathbf{h}_i]}^k$ to be the i-th simulated ciphertext (where $\overline{[\mathbf{h}_i]}^k \in \mathbb{G}^k$ denotes the vector consisting of the first k entries of $[\mathbf{h}_i]$) and proceeds the simulation with $\mathbf{X}_i := \mathsf{h}_0(\mathbf{W}[\mathbf{h}_i])$. In case $\mathcal{B}$ was given a real $\mathcal{U}_{k^2+1,k}$-MDDH tuple, that is there exist $\mathbf{s}_i \in \mathbb{Z}_p^k$ such that $[\mathbf{h}_i] = [\mathbf{B}]\mathbf{s}_i$ for all $i \in \{1, \ldots, Q_{\text{sim}}\}$, the adversary $\mathcal{B}$ simulates game $\mathbf{G}_{1.2}$.

In case the adversary was given a random challenge instead, the vectors $\mathbf{h}_i$ are distributed uniformly over $\mathbb{Z}_p^{k^2+1}$ and the adversary simulates a game statistically close to $\mathbf{G}_2$ (as $\mathbf{r} \leftarrow_R \mathbb{Z}_p^k$ is distributed equally to $\overline{\mathbf{Br}}^k$ for $\mathbf{r} \leftarrow \mathbb{Z}_p^k$, $\mathbf{B} \leftarrow_R Z_p^{(k^2+1)\times k}$).

Finally, by Theorem 6 and Theorem 5, together with the previous observations we obtain an adversary $\mathcal{B}'$ such that $T(\mathcal{B}') \approx T(\mathcal{A}) + (Q_{\text{ver}} + Q_{\text{sim}}) \cdot \mathsf{poly}(\lambda)$ and

$$|\varepsilon_2 - \varepsilon_1| \leq \mathrm{Adv}^{\text{mddh}}_{\mathbb{G},\mathcal{D}_{k^2+1,k},\mathcal{B}'}(\lambda) + 2^{-\Omega(\lambda)}.$$

Note that in order to prove this transition we require that in the definition of constrained soundness the simulation oracle returns random challenges (otherwise we would not be able to embed the $\mathcal{D}_{k^2+1,k}$-MDDH challenge into simulation queries). This is another reason why we cannot directly employ the notion of benign proof systems [Hof17].

Transition $\mathbf{G}_2 \rightsquigarrow \mathbf{G}_3$: As h_0 is universal, we can employ the Leftover Hash Lemma (Theorem 15) to switch $(\mathsf{h}_0, \mathsf{h}_0([\mathbf{u}]))$ to $(\mathsf{h}_0, \mathbf{U})$ in all simulation queries, where $\mathbf{U} \leftarrow_R \mathbb{Z}_p^{k\times k}$. A hybrid argument yields

$$|\varepsilon_2 - \varepsilon_3| \leq Q_{\text{sim}}/p.$$

Game $\mathbf{G}_3$: We show that $\varepsilon_3 \leq Q_{\text{ver}} \cdot \mathrm{uncert}^{\text{snd}}_{\mathcal{A}}(\lambda)$, where Q_{ver} is the number of queries to $\mathcal{O}_{\text{ver}}$ and $\mathrm{uncert}^{\text{snd}}_{\mathcal{A}}(\lambda)$ describes the uncertainty of the predicates provided by the adversary as described in Theorem 29.

We use a hybrid argument over the Q_{ver} queries to $\mathcal{O}_{\text{ver}}$. To that end, we introduce games $\mathbf{G}_{3.i}$ for $i = 0, \ldots, Q_{\text{ver}}$, defined as $\mathbf{G}_3$ except that for its first i queries $\mathcal{O}_{\text{ver}}$ answers $\perp$ on any query $([\mathbf{t}], [\pi], \mathsf{pred})$ with $[\mathbf{t}] \notin \mathcal{L}^{\text{snd}}$. We have $\varepsilon_3 = \varepsilon_{3.0}$, $\varepsilon_{3.Q_{\text{ver}}} = 0$ and we show that for all $i = 0, \ldots, Q_{\text{ver}} - 1$ it holds

$$|\varepsilon_{3.i} - \varepsilon_{3.(i+1)}| \leq \Pr_{K\in\mathcal{K}}[\mathsf{pred}_{i+1}(K) = 1] + 2^{-\Omega(\lambda)},$$

where pred_{i+1} is the predicate contained in the $(i+1)$-st query to $\mathcal{O}_{\text{ver}}$.

Games $\mathbf{G}_{3.i}$ and $\mathbf{G}_{3.(i+1)}$ behave identically on the first i queries to $\mathcal{O}_{\text{ver}}$. An adversary can only distinguish between the two, if it manages to provide a valid $(i+1)$-st query $([\mathbf{t}], [\pi], \mathsf{pred})$ to $\mathcal{O}_{\text{ver}}$ with $[\mathbf{t}] \notin \mathcal{L}^{\text{snd}}$. In the following we bound the probability of this happening.

From queries to $\mathcal{O}_{\text{sim}}$ and the first i queries to $\mathcal{O}_{\text{ver}}$ the adversary can only learn valid tuples $([\mathbf{t}], [\pi], [\kappa])$ with $[\mathbf{t}] \in \mathcal{L}^{\text{snd}}$. As explained in the beginning, such combined proofs reveal nothing about $\mathbf{K_y}$ beyond what is already revealed in the public key, as either $[\mathbf{t}] = [\mathbf{Ar}]$ for an $\mathbf{r} \in \mathbb{Z}_p^k$ and $\mathbf{y} = \mathsf{h}_1([\mathbf{K_y t}]) = \mathsf{h}_1([\mathbf{K_y A}]\mathbf{r})$ or $[\mathbf{t}] = [\mathbf{A}_0\mathbf{r}]$ and $[\pi, \mathbf{K}] = [\mathbf{A}_0](\mathbf{X} + \mathbf{r} \cdot \mathbf{y}^\top)$. In the former case $\mathbf{y}$ itself reveals no more about $\mathbf{K_y}$ than the public key, while in the latter case $\mathbf{y}$ is hidden by the fully randomized $\mathbf{X}$.

For any $[\mathbf{t}] \notin \mathcal{L}^{\text{snd}}$, $\mathbf{y} = \mathsf{h}_1([\mathbf{K_y t}])$ computed by $\mathcal{O}_{\text{ver}}$ is distributed statistically close to uniform from the adversaries point of view because of the following. First we can replace $\mathbf{K_y}$ by $\mathbf{K_y} + \mathbf{U}(\mathbf{A}^\perp)^\top$ for $\mathbf{U} \leftarrow_R \mathbb{Z}_p^{(k+1)\times k}$ and $\mathbf{A}^\perp \in \text{orth}(\mathbf{A})$ as both are distributed identically. By our considerations, this extra term is neither revealed through the public key nor through the previous queries to $\mathcal{O}_{\text{sim}}$ and $\mathcal{O}_{\text{ver}}$.

Now Theorem 15 (Leftover Hash Lemma) implies that the distribution of $\mathbf{y}$ is statistically close to uniform as desired. Since $[\mathbf{t}] \notin \text{span}([\mathbf{A}_0])$ we have $[\mathbf{a}] := \underline{[\mathbf{t}]} - \underline{[\mathbf{A}_0]}\overline{\mathbf{A}_0}^{-1}\overline{[\mathbf{t}]} \neq 0$. In other words, there exists an $i \in \{1, \ldots, k\}$ such that $\overline{[\mathbf{a}]_i \neq 0}$ and thus $[\mathbf{a}]_i \cdot \mathbf{y}_i$ is distributed uniformly at random from the adversary's point of view. Recall that the key $[\kappa]$ is computed as

$$[\kappa] := \text{trace}\left(\underline{\mathbf{A}_0}\overline{\mathbf{A}_0}^{-1}[\pi^\star] + \underbrace{\left(\underline{[\mathbf{t}]} - \underline{\mathbf{A}_0}\overline{\mathbf{A}_0}^{-1}\overline{[\mathbf{t}]}\right)}_{\neq 0} \cdot \mathbf{y}^\top\right)$$

by $\mathcal{O}_{\text{ver}}$, so in particular $[\kappa]$ consists of $[\mathbf{a}]_i \cdot \mathbf{y}_i$ plus independent summands and is thus distributed uniformly at random over $\mathbb{Z}_p$ as well.

Altogether, we obtain

$$\varepsilon_3 \leq Q_{\text{ver}} \cdot \text{uncert}^{\text{snd}}_{\mathcal{A}}(\lambda) + Q_{\text{ver}} \cdot 2^{-\Omega(\lambda)}.$$

□

Extensibility to a Three-Way Or-Proof. In the following we prove that the proof system in Fig. 3.1 satisfies $\mathcal{L}^{\text{ext}}$-extensibility (see Theorem 31). This will enable us to employ soundness, even in the presence of many simulated proofs for statements in

$$\mathcal{L}^{\text{ext}} := \text{span}([\mathbf{A}]) \cup \text{span}([\mathbf{A}_0]) \cup \text{span}([\mathbf{A}_1]).$$

In this section we implicitly assume all algorithms to have access to $\mathsf{pars}_{\mathsf{PS}'} := (\mathsf{pars}_{\mathsf{PS}}, [\mathbf{A}_1])$.

We describe a proof system PS' for $\mathcal{L}$ in Fig. 3.4. We prove that it is $\mathcal{L}^{\text{snd}}$-indistinguishable to PS in Theorem 34, and prove that it complies with constrained $\mathcal{L}^{\text{ext}}$-soundess in Theorem 35.

Lemma 34 ($\mathcal{L}^{\text{snd}}$-indistinguishability)**.** *The proof systems* PS *and* PS' *described in Fig. 3.1 and Fig. 3.4, resp., are* $\mathcal{L}^{\text{snd}}$*-indistinguishable. That is, for every (unbounded) adversary* $\mathcal{A}$ *we have* $\text{Adv}^{\text{PS-ind}}_{\mathcal{L}^{\text{snd}},\mathsf{PS},\mathsf{PS}',\mathcal{A}}(\lambda) = 2^{-\Omega(\lambda)}$.

$\underline{\mathsf{PGen}'(1^\lambda)}$:
$\mathbf{K}_\mathbf{X}, \mathbf{K}'_\mathbf{X} \leftarrow_R \mathbb{Z}_p^{(k^2+1)\times 2k}$
$\mathbf{K}_\mathbf{y}, \mathbf{K}'_\mathbf{y} \leftarrow_R \mathbb{Z}_p^{(k+1)\times 2k}$
$\mathbf{A}^\perp \in \mathrm{orth}(\mathbf{A})$
$\mathsf{ppk} := ([\mathbf{K}_\mathbf{X}\mathbf{A}], [\mathbf{K}_\mathbf{y}\mathbf{A}])$
$\mathsf{psk} := (\mathbf{K}_\mathbf{X}, \mathbf{K}_\mathbf{y}, \mathbf{K}'_\mathbf{X}, \mathbf{K}'_\mathbf{y}, \mathbf{A}^\perp)$
return $(\mathsf{ppk}, \mathsf{psk})$

$\underline{\widetilde{\mathsf{PVer}}(\mathsf{ppk}, \mathsf{psk}, [\mathbf{t}], [\pi^\star])}$:
if $[\mathbf{t}]^\top \mathbf{A}^\perp = [\mathbf{0}]$
  $\mathbf{X} := \mathsf{h}_0(\mathbf{K}_\mathbf{X}[\mathbf{t}]) \in \mathbb{Z}_p^{k\times k}$
  $\mathbf{y} := \mathsf{h}_1(\mathbf{K}_\mathbf{y}[\mathbf{t}]) \in \mathbb{Z}_p^{k}$
else
  $\mathbf{X} := \mathsf{h}_0(\mathbf{K}'_\mathbf{X}[\mathbf{t}]) \in \mathbb{Z}_p^{k\times k}$
  $\mathbf{y} := \mathsf{h}_1(\mathbf{K}'_\mathbf{y}[\mathbf{t}]) \in \mathbb{Z}_p^{k}$
$[\pi] := \overline{[\mathbf{A}_0]} \cdot \mathbf{X} + \overline{[\mathbf{t}]} \cdot \mathbf{y}^\top \in \mathbb{G}^{k\times k}$
$[\mathbf{K}] := \underline{[\mathbf{A}_0]} \cdot \mathbf{X} + \underline{[\mathbf{t}]} \cdot \mathbf{y}^\top \in \mathbb{G}^{k\times k}$
$[\kappa] := \overline{\mathrm{trace}}([\mathbf{K}]) \in \mathbb{G}$
if $[\pi] = [\pi^\star]$ **return** $(1, [\kappa])$
else return $(0, \perp)$

$\underline{\mathsf{PPrv}'(\mathsf{ppk}, [\mathbf{t}], \mathbf{r})}$:
$\mathbf{X} := \mathsf{h}_0([\mathbf{K}_\mathbf{X}\mathbf{A}]\mathbf{r}) \in \mathbb{Z}_p^{k\times k}$
$\mathbf{y} := \mathsf{h}_1([\mathbf{K}_\mathbf{y}\mathbf{A}]\mathbf{r}) \in \mathbb{Z}_p^{k}$
$[\pi] := \overline{[\mathbf{A}_0]} \cdot \mathbf{X} + \overline{[\mathbf{t}]} \cdot \mathbf{y}^\top \in \mathbb{G}^{k\times k}$
$[\mathbf{K}] := \underline{[\mathbf{A}_0]} \cdot \mathbf{X} + \underline{[\mathbf{t}]} \cdot \mathbf{y}^\top \in \mathbb{G}^{k\times k}$
$[\kappa] := \overline{\mathrm{trace}}([\mathbf{K}]) \in \mathbb{G}$
return $([\pi], [\kappa])$

$\underline{\mathsf{PSim}'(\mathsf{ppk}, \mathsf{psk}, [\mathbf{t}])}$:
if $[\mathbf{t}]^\top \mathbf{A}^\perp = [\mathbf{0}]$
  $\mathbf{X} := \mathsf{h}_0(\mathbf{K}_\mathbf{X}[\mathbf{t}]) \in \mathbb{Z}_p^{k\times k}$
  $\mathbf{y} := \mathsf{h}_1(\mathbf{K}_\mathbf{y}[\mathbf{t}]) \in \mathbb{Z}_p^{k}$
else
  $\mathbf{X} := \mathsf{h}_0(\mathbf{K}'_\mathbf{X}[\mathbf{t}]) \in \mathbb{Z}_p^{k\times k}$
  $\mathbf{y} := \mathsf{h}_1(\mathbf{K}'_\mathbf{y}[\mathbf{t}]) \in \mathbb{Z}_p^{k}$
$[\pi] := \overline{[\mathbf{A}_0]} \cdot \mathbf{X} + \overline{[\mathbf{t}]} \cdot \mathbf{y}^\top \in \mathbb{G}^{k\times k}$
$[\mathbf{K}] := \underline{[\mathbf{A}_0]} \cdot \mathbf{X} + \underline{[\mathbf{t}]} \cdot \mathbf{y}^\top \in \mathbb{G}^{k\times k}$
$[\kappa] := \overline{\mathrm{trace}}([\mathbf{K}]) \in \mathbb{G}$
return $([\pi], [\kappa])$

Figure 3.4: $\mathcal{L}^{\mathrm{ext}}$-qualified proof system PS' for $\mathcal{L}$.

Proof. PS only differs from PS' for statements $[\mathbf{t}] \notin \mathcal{L}$, and since we are interested in $\mathcal{L}^{\mathrm{snd}}$-indistinguishability, it suffices to consider $[\mathbf{t}] \in \mathrm{span}([\mathbf{A}_0])$. To argue that the two proof systems are statistically indistinguishable for statements $[\mathbf{t}] \in \mathrm{span}([\mathbf{A}_0])$, we use the following.

First, $\mathbf{K}_\mathbf{X}$ and $\mathbf{K}_\mathbf{X} + \mathbf{U}(\mathbf{A}^\perp)^\top$ are identically distributed for $\mathbf{K}_\mathbf{X} \leftarrow_R \mathbb{Z}_p^{(k^2+1)\times 2k}$, $\mathbf{U} \leftarrow_R \mathbb{Z}_p^{(k^2+1)\times k}$, and $\mathbf{A}^\perp \in \mathrm{orth}(\mathbf{A})$. Note that the extra term $\mathbf{U}(\mathbf{A}^\perp)^\top$ does not show up in either the ppk or in oracle-queries of the $\mathcal{L}^{\mathrm{snd}}$-indistinguishability game for statements $[\mathbf{t}] \in \mathrm{span}([\mathbf{A}])$ since for all $\mathbf{t} \in \mathrm{span}(\mathbf{A})$ we have $(\mathbf{K}_\mathbf{X} + \mathbf{U}(\mathbf{A}^\perp)^\top)\mathbf{t} = \mathbf{K}_\mathbf{X}\mathbf{t}$.

Further, for all $\mathbf{t} \in \mathrm{span}(\mathbf{A}_0)$, $\mathbf{A}_0^\perp \in \mathrm{orth}(\mathbf{A}_0)$, we have

$$\mathbf{U}(\mathbf{A}^\perp)^\top \mathbf{t} = \left(\mathbf{U}(\mathbf{A}^\perp)^\top + \mathbf{U}_0(\mathbf{A}_0^\perp)^\top\right)\mathbf{t},$$

where $\mathbf{U}, \mathbf{U}_0 \leftarrow_R \mathbb{Z}_p^{(k^2+1)\times k}$.

With probability $1 - 2^{-\Omega(\lambda)}$ over the choices of $\mathbf{A}, \mathbf{A}_0$ the vectors in $\mathbf{A}^\perp$ and $\mathbf{A}_0^\perp$ together form a basis of $\mathbb{Z}_p^{2k}$, in which case the matrix $\mathbf{U}(\mathbf{A}^\perp)^\top + \mathbf{U}_0(\mathbf{A}_0^\perp)^\top$ is distributed uniformly random over $\mathbb{Z}_p^{(k^2+1)\times 2k}$.

In conclusion, with overwhelming probability over the choice of the public parameters we obtain that for all $\mathbf{t} \in \mathrm{span}(\mathbf{A}_0)$, $(\mathbf{K}_\mathbf{X}\mathbf{A}, \mathbf{K}_\mathbf{X}\mathbf{t})$ is identically distributed to

$(\mathbf{K_X A}, \mathbf{K'_X t})$, where $\mathbf{K'_X} \leftarrow_R \mathbb{Z}_p^{(k^2+1)\times 2k}$ is chosen uniformly at random, independently of $\mathbf{K_X}$.

By the same reasoning we obtain that for all $\mathbf{t} \in \mathrm{span}(\mathbf{A}_0)$, for an independently and uniformly at random chosen key $\mathbf{K'_y} \leftarrow_R \mathbb{Z}_p^{(k+1)\times 2k}$ the tuples $(\mathbf{K_y A}, \mathbf{K_y t})$ and $(\mathbf{K_y A}, \mathbf{K'_y t})$ are statistically close with overwhelming probability.

This proves the lemma. □

The techniques used in the following to prove constrained $\mathcal{L}^{\mathrm{ext}}$-soundness of PS' are very similar to the ones presented in the proof of Theorem 33.

Lemma 35 (Constrained $\mathcal{L}^{\mathrm{ext}}$-soundness of PS')**.** *If the $\mathcal{D}_{k^2+1,k}$-MDDH assumption holds in $\mathbb{G}$ and h_0 and h_1 are universal hash functions, then the proof system described in Fig. 3.4 complies with constrained $\mathcal{L}^{\mathrm{ext}}$-soundness. Namely, for any adversary $\mathcal{A}$ against $\mathcal{L}^{\mathrm{ext}}$-soundness, there exists an adversary $\mathcal{B}$ such that $T(\mathcal{B}) \approx T(\mathcal{A}) + (Q_{\mathrm{sim}} + Q_{\mathrm{ver}}) \cdot \mathsf{poly}(\lambda)$ and*

$$\begin{aligned}\mathrm{Adv}^{\mathrm{snd}}_{\mathcal{L}^{\mathrm{ext}},\mathsf{PS}',\mathcal{A}}(\lambda) \leq\ & \mathrm{Adv}^{\mathrm{mddh}}_{\mathbb{G},\mathcal{B},\mathcal{D}_{k^2+1,k}}(\lambda) + Q_{\mathrm{ver}} \cdot \mathrm{uncert}^{\mathrm{snd}}_{\mathcal{A}}(\lambda) \\ & + (Q_{\mathrm{sim}} + Q_{\mathrm{ver}} + 1) \cdot 2^{-\Omega(\lambda)},\end{aligned}$$

where Q_{sim}, Q_{ver} are the number of calls to $\mathcal{O}_{\mathrm{sim}}$ and $\mathcal{O}_{\mathrm{ver}}$ respectively, $\mathrm{uncert}^{\mathrm{snd}}_{\mathcal{A}}(\lambda)$ *describes the uncertainty of the predicates provided by $\mathcal{A}$ and* poly *is a polynomial function independent of $T(\mathcal{A})$.*

Proof. We prove the $\mathcal{L}^{\mathrm{ext}}$-soundness of PS' via a series of games, described in Fig. 3.5. The proof is very similar to the proof of Theorem 33.

We start with the $\mathcal{L}^{\mathrm{ext}}$-constrained soundness game, which we refer to as game $\mathbf{G}$. In the following we want to bound the probability

$$\varepsilon := \mathrm{Adv}^{\mathrm{snd}}_{\mathsf{PS},\mathcal{A}}(\lambda).$$

We denote the probability that the adversary $\mathcal{A}$ wins the game $\mathbf{G}_i$ by

$$\varepsilon_i := \mathrm{Adv}_{\mathbf{G}_i,\mathcal{A}}(\lambda).$$

We omit the proof of the game transitions $\mathbf{G} \rightsquigarrow \mathbf{G}_0$ and $\mathbf{G}_0 \rightsquigarrow \mathbf{G}_1$, as they follow the proof of Theorem 33 almost verbatim.

Transition $\mathbf{G}_1 \rightsquigarrow \mathbf{G}_2$: This transition is similar to the transition $\mathbf{G}_1 \rightsquigarrow \mathbf{G}_2$ of Theorem 33.Again, we proceed via a series of intermediary games $\mathbf{G}_{1.1}, \mathbf{G}_{1.2}$ and $\mathbf{G}_{1.3}$.

In game $\mathbf{G}_{1.1}$ we change the way $\mathbf{K'_X}$ is computed, namely we replace $\mathbf{K'_X}$ by $\mathbf{K'_X} + \mathbf{U}_0 \cdot (\mathbf{A}_0^{\perp})^{\top} + \mathbf{U}_1 \cdot (\mathbf{A}_1^{\perp})^{\top}$ for $\mathbf{U}_0, \mathbf{U}_1 \leftarrow_R \mathbb{Z}_p^{(k^2+1)\times k}$, $\mathbf{A}_0^{\perp} \in \mathrm{orth}(\mathbf{A}_0)$ and $\mathbf{A}_1^{\perp} \in \mathrm{orth}(\mathbf{A}_1)$. As both are distributed equally we obtain

$$\varepsilon_{1.1} = \varepsilon_1.$$

In game $\mathbf{G}_{1.2}$ we change the way $\mathbf{X}$ is computed, namely we additionally draw $\mathbf{B} \leftarrow_R \mathbb{Z}_p^{(k^2+1)\times k}$ and $\mathbf{W}_0, \mathbf{W}_1 \leftarrow_R \mathbb{Z}_p^{(k^2+1)\times(k^2+1)}$ during set-up. For a simulation query from now we first choose $b \leftarrow_R \{0,1\}$ to decide whether to

#	**sim. X for** $[\mathbf{t}] \in \mathcal{L}^{\text{ext}} \backslash \mathcal{L}$	**ver.** $[\mathbf{K}]$ **for** $[\mathbf{t}] \notin \mathcal{L}$	**game knows**	**remark**
$\mathbf{G}_0$	$\mathbf{X} := \mathsf{h}_0(\mathbf{K}'_{\mathbf{X}}[\mathbf{t}])$	$\underline{[\mathbf{A}_0]} \cdot \mathbf{x} + \underline{[\mathbf{t}]} \cdot \mathbf{y}^\top$	$\mathbf{A}$	$\mathcal{L}^{\text{ext}}$-soundn. game w/o *lose*
$\mathbf{G}_1$	$\mathbf{X} := \mathsf{h}_0(\mathbf{K}'_{\mathbf{X}}[\mathbf{t}])$	$\underline{\mathbf{A}_0}\overline{\mathbf{A}_0}^{-1}\left([\pi^\star] - \overline{[\mathbf{t}]} \cdot \mathbf{y}^\top\right) + \underline{[\mathbf{t}]} \cdot \mathbf{y}^\top$	$\mathbf{A}$, $\mathbf{A}_0$	win. chances increase
$\mathbf{G}_2$	$\mathbf{u} \leftarrow_R \mathbb{Z}_p^{k^2+1}$, $\mathbf{X} := \mathsf{h}_0([\mathbf{u}])$	$\underline{\mathbf{A}_0}\overline{\mathbf{A}_0}^{-1}\left([\pi^\star] - \overline{[\mathbf{t}]} \cdot \mathbf{y}^\top\right) + \underline{[\mathbf{t}]} \cdot \mathbf{y}^\top$	$\mathbf{A}$, $\mathbf{A}_0$, $\mathbf{A}_1$	$\mathcal{D}_{k^2+1,k}$-MDDH
$\mathbf{G}_3$	$\mathbf{X} \leftarrow_R \mathbb{Z}_p^{k \times k}$	$\underline{\mathbf{A}_0}\overline{\mathbf{A}_0}^{-1}\left([\pi^\star] - \overline{[\mathbf{t}]} \cdot \mathbf{y}^\top\right) + \underline{[\mathbf{t}]} \cdot \mathbf{y}^\top$	$\mathbf{A}$, $\mathbf{A}_0$, $\mathbf{A}_1$	Theorem 15 (LOHL)

Figure 3.5: Overview of the proof of $\mathcal{L}^{\text{ext}}$-constrained soundness of PS. The first column shows how $\mathbf{X}$ is computed for queries to $\mathcal{O}_{\text{sim}}$ with $[\mathbf{t}] \in \mathcal{L}^{\text{ext}} \backslash \mathcal{L}$. The second column shows how the pre-key $[\mathbf{K}]$ computed by the verifier in queries to $\mathcal{O}_{\text{ver}}$ for $[\mathbf{t}] \notin \mathcal{L}$. Recall that the key is computed as $[\kappa] := \text{trace}([\mathbf{K}])$. The third column "**game knows**" gives an overview of which non-public information need to be known by the game respective to $\mathbf{A}$, $\mathbf{A}_0$ and $\mathbf{A}_1$.

return $[\mathbf{t}] \in \mathcal{L}^{\text{snd}} \backslash \mathcal{L}$ or $[\mathbf{t}] \in \mathcal{L}^{\text{ext}} \backslash \mathcal{L}^{\text{snd}}$. This yields the correct distribution as $|\mathcal{L}^{\text{snd}} \backslash \mathcal{L}| = |\mathcal{L}^{\text{ext}} \backslash \mathcal{L}^{\text{snd}}|$. We then choose $\mathbf{r} \leftarrow_R \mathbb{Z}_p^k$ and set $[\mathbf{t}] := [\mathbf{A}_b] \cdot \overline{\mathbf{Br}}^k$ (where $\overline{\mathbf{Br}}^k$ denotes the vector comprising the upper k entries of $\mathbf{Br}$. Further, we set $\mathbf{X}$ to be $\mathbf{X} := \mathsf{h}_0(\mathbf{W}_b[\mathbf{Br}])$. To see that game $\mathbf{G}_{1.1}$ and game $\mathbf{G}_{1.2}$ are statistically close, note that $\mathbf{r} \leftarrow_R \mathbb{Z}_p^k$ is distributed equally to $\overline{\mathbf{Br}}^k$ for $\mathbf{r} \leftarrow \mathbb{Z}_p^k$, $\mathbf{B} \leftarrow_R Z_p^{(k^2+1)\times k}$. Further, with overwhelming probablity over the choices of $\mathbf{A}_0$, $\mathbf{A}_1$ the matrices $(\mathbf{A}_1^\perp)^\top \cdot \mathbf{A}_0 \in \mathbb{Z}_p^{k\times k}$ and $(\mathbf{A}_0^\perp)^\top \cdot \mathbf{A}_1 \in \mathbb{Z}_p^{k\times k}$ are invertible, which implies that $(\mathbf{K}'_{\mathbf{X}} + \mathbf{U}_0 \cdot (\mathbf{A}_0^\perp)^\top + \mathbf{U}_1 \cdot (\mathbf{A}_1^\perp)^\top) \cdot \mathbf{A}_0 = (\mathbf{K}'_{\mathbf{X}} + \mathbf{U}_1 \cdot (\mathbf{A}_1^\perp)^\top) \cdot \mathbf{A}_0$ is distributed uniformly random over $\mathbb{Z}_p^{(k^2+1)\times k}$ and stochastic independent of $(\mathbf{K}'_{\mathbf{X}} + \mathbf{U}_0 \cdot (\mathbf{A}_0^\perp)^\top + \mathbf{U}_1 \cdot (\mathbf{A}_1^\perp)^\top) \cdot \mathbf{A}_1 = (\mathbf{K}'_{\mathbf{X}} + \mathbf{U}_0 \cdot (\mathbf{A}_0^\perp)^\top) \cdot \mathbf{A}_1$. Thus switching between $(\mathbf{K}'_{\mathbf{X}} + \mathbf{U}_{1-\beta} \cdot (\mathbf{A}_{1-\beta}^\perp)^\top) \cdot \mathbf{A}_\beta$ and $\mathbf{W}_\beta \cdot \mathbf{B} \cdot (\overline{\mathbf{B}}^k)^{-1}$ for $\beta \in \{0,1\}$ is statistically indistinguishable to $\mathcal{A}$ (where $\overline{\mathbf{B}}^k \in \mathbb{Z}_p^{k\times k}$ denotes the upper square matrix of $\mathbf{B}$). This yields

$$\begin{aligned}
\mathbf{X} &= \mathsf{h}_0(\mathbf{K}'_{\mathbf{X}}[\mathbf{t}_i]) \\
&\equiv_s \mathsf{h}_0((\mathbf{K}'_{\mathbf{X}} + \mathbf{U}_{1-b} \cdot (\mathbf{A}_{1-b}^\perp)^\top) \cdot [\mathbf{A}_b \overline{\mathbf{Br}}^k]) \\
&\equiv_s \mathsf{h}_0(\mathbf{W}_b \cdot \mathbf{B} \cdot (\overline{\mathbf{B}}^k)^{-1} \cdot [\overline{\mathbf{Br}}^k]) \\
&= \mathsf{h}_0(\mathbf{W}_b \cdot [\mathbf{Br}]).
\end{aligned}$$

and thus

$$|\varepsilon_{1.2} - \varepsilon_{1.1}| \leq 2^{-\Omega(\lambda)}.$$

Next, we reverse transition $\mathbf{G}_1 \rightsquigarrow \mathbf{G}_{1.2}$, that is we choose $\mathbf{K}'_{\mathbf{X}} \leftarrow_R \mathbb{Z}_p^{(k^k+1)times2k}$ again. This change does not show up, as from game $\mathbf{G}_{1.2}$ on $\mathbf{K}'_{\mathbf{X}}$ is not employed anymore. We thus have $\varepsilon_{1.3} = \varepsilon_{1.2}$.

Next we want to bound transition $\mathbf{G}_{1.3} \rightsquigarrow \mathbf{G}_2$ by bounding the advantage of an adversary $\mathcal{A}$ distinguishing between $\mathbf{G}_{1.3}$ and $\mathbf{G}_2$. To this end let $([\mathbf{B}], [\mathbf{h}_1, \ldots, \mathbf{h}_{Q_{\text{sim}}}])$ be a Q_{sim}-fold $\mathcal{U}_{k^2+1,k}$-MDDH challenge. First, $\mathcal{B}$ picks $\mathbf{A}, \mathbf{A}_0, \mathbf{A}_1$ as described in Section 3.2 and further draws $\mathbf{K}_{\mathbf{X}}, \mathbf{K}'_{\mathbf{X}} \leftarrow_R \mathbb{Z}_p^{(k^2+1)\times 2k}$ and $\mathbf{K}_{\mathbf{y}}, \mathbf{K}'_{\mathbf{y}} \leftarrow_R \mathbb{Z}_p^{(k+1)\times 2k}$. Next, $\mathcal{B}$ send $[\mathbf{A}]$, $[\mathbf{A}_0]$, $[\mathbf{A}_1]$ and $\mathsf{ppk} := ([\mathbf{K}_{\mathbf{X}}\mathbf{A}], [\mathbf{K}_{\mathbf{y}}\mathbf{A}])$ to $\mathcal{A}$. Further, $\mathcal{B}$ chooses $\mathbf{W}_0, \mathbf{W}_1 \leftarrow \mathbb{Z}_p^{(k^2+1)\times(k^2+1)}$ at random.

Verification queries are answered by $\mathcal{B}$ according to the verification oracle $\mathcal{O}_{\text{ver}}$ in game $\mathbf{G}_1$.

On the i-th query to $\mathcal{O}_{\text{sim}}$, for all $i \in [Q_{\text{sim}}]$, the adversary $\mathcal{B}$ first choses $b \leftarrow_R \{0,1\}$ to decide whether to return $[\mathbf{t}] \in \mathcal{L}^{\text{snd}} \backslash \mathcal{L}$ or $[\mathbf{t}] \in \mathcal{L}^{\text{ext}} \backslash \mathcal{L}^{\text{snd}}$. The adversary then sets the i-th simulation element to equal $[\mathbf{t}_i] := \mathbf{A}_b \overline{[\mathbf{h}_i]}^k$ (where $\overline{[\mathbf{h}_i]}^k \in \mathbb{G}^k$ denotes the upper k entries of $[\mathbf{h}_i]$) and continues the simulation with $\mathbf{X}_i := \mathsf{h}_0(\mathbf{W}_b[\mathbf{h}_i])$.

Now in case of a real $\mathcal{U}_{k^2+1,k}$-MDDH challenge, the adversary $\mathcal{B}$ simulates game $\mathbf{G}_{1.3}$.

In case the adversary was given a random challenge, the vectors $\mathbf{h}_i$ are distributed uniformly over $\mathbb{Z}_p^{k^2+1}$ and the adversary simulates a game statistically close to $\mathbf{G}_2$.

Finally, by Theorem 7 and Theorem 5 together with our previous observations, we obtain an adversary $\mathcal{B}'$ such that $T(\mathcal{B}') \approx T(\mathcal{A}) + (Q_{\text{ver}} + Q_{\text{sim}}) \cdot \mathsf{poly}(\lambda)$ and

$$|\varepsilon_2 - \varepsilon_1| \leq \text{Adv}^{\text{mddh}}_{\mathbb{G},\mathcal{D}_{k^2+1,k},\mathcal{B}'}(\lambda) + 2^{-\Omega(\lambda)}.$$

Transition $\mathbf{G}_2 \rightsquigarrow \mathbf{G}_3$: As h_0 is universal, we can employ the Leftover Hash Lemma (Theorem 15) to switch $(\mathsf{h}_0, \mathsf{h}_0([\mathbf{u}]))$ to $(\mathsf{h}_0, \mathbf{U})$ in all simulation queries, where $\mathbf{U} \leftarrow_R \mathbb{Z}_p^{k\times k}$. A hybrid argument yields

$$|\varepsilon_2 - \varepsilon_3| \leq Q_{\text{sim}}/p.$$

Game $\mathbf{G}_3$: We show that $\varepsilon_3 \leq Q_{\text{ver}} \cdot \text{uncert}^{\text{snd}}_{\mathcal{A}}(\lambda)$, where Q_{ver} is the number of queries to $\mathcal{O}_{\text{ver}}$ and $\text{uncert}^{\text{snd}}_{\mathcal{A}}(\lambda)$ describes the uncertainty of the predicates provided by the adversary as described in Theorem 29.

Similar to game $\mathbf{G}_3$ in Theorem 33 we use a hybrid argument over the Q_{ver} queries to $\mathcal{O}_{\text{ver}}$. To that end, we introduce games $\mathbf{G}_{3.i}$ for $i = 0, \ldots, Q_{\text{ver}}$, defined as $\mathbf{G}_3$ except that for its first i queries $\mathcal{O}_{\text{ver}}$ answers $\perp$ on any query $([\mathbf{t}], [\pi], \mathsf{pred})$ with $[\mathbf{t}] \notin \mathcal{L}^{\text{ext}}$. We have $\varepsilon_3 = \varepsilon_{3.0}$, $\varepsilon_{3.Q_{\text{ver}}} = 0$ and we show that for all $i = 0, \ldots, Q_{\text{ver}} - 1$ it holds

$$|\varepsilon_{3.i} - \varepsilon_{3.(i+1)}| \leq \Pr_{K\in\mathcal{K}}[\mathsf{pred}_{i+1}(K) = 1] + 2^{-\Omega(\lambda)},$$

where pred_{i+1} is the predicate contained in the $(i+1)$-st query to $\mathcal{O}_{\text{ver}}$.

Games $\mathbf{G}_{3.i}$ and $\mathbf{G}_{3.(i+1)}$ behave identically on the first i queries to $\mathcal{O}_{\text{ver}}$. An adversary can only distinguish between the two, if it manages to provide a valid $(i+1)$-st query $([\mathbf{t}], [\pi], \mathsf{pred})$ to $\mathcal{O}_{\text{ver}}$ with $[\mathbf{t}] \notin \mathcal{L}^{\text{ext}}$. In the following we bound the probability of this happening.

From queries to $\mathcal{O}_{\text{sim}}$ and the first i queries to $\mathcal{O}_{\text{ver}}$ the adversary can only learn valid tuples $([\mathbf{t}], [\pi], [\kappa])$ with $[\mathbf{t}] \in \mathcal{L}^{\text{ext}}$. Such combined proofs reveal nothing about $\mathbf{K}'_{\mathbf{y}}$ beyond $[\mathbf{K}'_{\mathbf{y}}\mathbf{A}_1]$. This holds as either $[\mathbf{t}] = [\mathbf{A}\mathbf{r}]$ for an $\mathbf{r} \in \mathbb{Z}_p^k$, in which case $\mathbf{K}'_{\mathbf{y}}$ is not employed at all, or $[\mathbf{t}] = [\mathbf{A}_0\mathbf{r}]$ and $[\pi, \mathbf{K}] = [\mathbf{A}_0](\mathbf{X} + \mathbf{r} \cdot \mathbf{y}^\top)$ and thus $\mathbf{y}$ (and therefore $\mathbf{K}'_{\mathbf{y}}$) is completely hidden by the randomized $\mathbf{X}$, or $[\mathbf{t}] = [\mathbf{A}_1\mathbf{r}]$, in which case only $[\mathbf{K}'_{\mathbf{y}}\mathbf{A}_1]$ is revealed.

For any $[\mathbf{t}] \notin \mathcal{L}^{\text{ext}}$, $\mathbf{y} = \mathsf{h}_1([\mathbf{K}'_{\mathbf{y}}\mathbf{t}])$ computed by $\mathcal{O}_{\text{ver}}$ is thus distributed statistically close to uniform from the adversary's point of view. Namely, we can replace $\mathbf{K}'_{\mathbf{y}}$ by $\mathbf{K}'_{\mathbf{y}} + \mathbf{U}(\mathbf{A}_1^\perp)^\top$ for $\mathbf{U} \leftarrow_R \mathbb{Z}_p^{(k+1)\times k}$ and $\mathbf{A}_1^\perp \in \text{orth}(\mathbf{A}_1)$ as both are distributed identically. By our considerations, this extra term is neither revealed through the public key nor through the previous queries to $\mathcal{O}_{\text{sim}}$ and $\mathcal{O}_{\text{ver}}$.

Now Theorem 15 (Leftover Hash Lemma) implies that the distribution of $\mathbf{y}$ is statistically close to uniform as desired. Since $[\mathbf{t}] \notin \text{span}([\mathbf{A}_0])$ we have $[\mathbf{a}] := \underline{[\mathbf{t}]} - \underline{[\mathbf{A}_0]}\overline{\mathbf{A}_0}^{-1}\overline{[\mathbf{t}]} \neq 0$. In other words, there exists an $i \in \{1, \ldots, k\}$ such that $[\mathbf{a}]_i \neq 0$ and thus $[\mathbf{a}]_i \cdot \mathbf{y}_i$ is distributed uniformly at random from the adversaries point of view. Recall that the key $[\kappa]$ is computed as

$$[\kappa] := \text{trace}\left(\underline{\mathbf{A}_0}\overline{\mathbf{A}_0}^{-1}[\pi^\star] + \underbrace{\left(\underline{[\mathbf{t}]} - \underline{\mathbf{A}_0}\overline{\mathbf{A}_0}^{-1}\overline{[\mathbf{t}]}\right)}_{\neq 0} \cdot \mathbf{y}^\top\right)$$

by $\mathcal{O}_{\text{ver}}$, so in particular $[\kappa]$ consists of $[\mathbf{a}]_i \cdot \mathbf{y}_i$ plus independent summands and is thus distributed uniformly over $\mathbb{Z}_p$ as well.

Altogether, we obtain

$$\varepsilon_3 \leq Q_{\text{ver}} \cdot \text{uncert}^{\text{snd}}_{\mathcal{A}}(\lambda) + Q_{\text{ver}} \cdot 2^{-\Omega(\lambda)}.$$

□

3.3 Key Encapsulation Mechanisms

One way of constructing a public-key encryption scheme is via a *key encapsulation mechanism (KEM)*. More precisely, by the results of [HK07], a KEM satisfying the security notion of *indistinguishability against constrained chosen-ciphertext attacks (IND-CCCA)* [HK07] together with an arbitrary authenticated symmetric encryption scheme, yields an IND-CCA secure hybrid encryption scheme.[1] Roughly speaking, the CCCA security experiment, in contrast to the CCA experiment, makes an additional requirement on decryption queries. Namely, in addition to the ciphertext, the adversary has to provide a predicate implying some partial knowledge about the key to be decrypted. The idea of hybrid encryption and the notion of a KEM was first formalized in [CS03].

Definition 36 (Key encapsulation mechanism)**.** A *key encapsulation mechanism* is a tuple of PPT algorithms $(\mathsf{KGen}, \mathsf{KEnc}, \mathsf{KDec})$ such that:

[1] The corresponding reduction is tight also in the multi-user and multi-ciphertext setting. Suitable (one-time) secure symmetric encryption schemes exist even unconditionally [HK07].

$\mathsf{KGen}(1^\lambda)$: On input of the security parameter λ in unary representation, generates a pair $(\mathsf{pk}, \mathsf{sk})$ of keys.

$\mathsf{KEnc}(\mathsf{pk})$: On input pk, returns a ciphertext C and a symmetric key $K \in \mathcal{K}(\lambda)$, where $\mathcal{K}(\lambda)$ is the key-space.

$\mathsf{KDec}(\mathsf{sk}, C)$: On input of the secret key sk and a ciphertext C returns a key $K \in \mathcal{K}(\lambda)$ or a special rejection symbol *bot*.

We say $(\mathsf{KGen}, \mathsf{KEnc}, \mathsf{KDec})$ is *perfectly correct*, if for all $\lambda \in \mathbb{N}$,

$$\Pr[\mathsf{KDec}(\mathsf{sk}, C) = K] = 1,$$

where $(\mathsf{pk}, \mathsf{sk}) \leftarrow_R \mathsf{Gen}(1^\lambda)$, $(K, C) \leftarrow_R \mathsf{KEnc}(\mathsf{pk})$ and the probability is taken over the random coins of Gen and KEnc.

Note that we always implicitly assume the secret key to contain the public key.

As mentioned above, for *constrained* chosen ciphertext security, the adversary has to have some knowledge about the key upfront in order to make a decryption query. As in [HK07] we will use a measure for the uncertainty left and require it to be negligible for every query, thereby only allowing decryption queries where the adversary has a high prior knowledge of the corresponding key. We now provide a formal definition.

Definition 37 (Multi-ciphertext IND-CCCA security)**.** For any key encapsulation mechanism $\mathsf{KEM} = (\mathsf{KGen}, \mathsf{KEnc}, \mathsf{KDec})$ and any stateful adversary $\mathcal{A}$, we define the following experiment:

$\underline{\mathrm{Exp}^{\mathrm{ccca}}_{\mathsf{KEM},\mathcal{A}}(\lambda)}$:	$\underline{\mathcal{O}_{\mathrm{enc}}}$:	$\underline{\mathcal{O}_{\mathrm{dec}}(\mathsf{pred}_i, C_i)}$:
$(pk, sk) \leftarrow_R \mathsf{KGen}(1^\lambda)$	$K_0 \leftarrow_R \mathcal{K}(\lambda)$	$K_i := \mathsf{KDec}(\mathsf{sk}, C_i)$
$b \leftarrow_R \{0,1\}$	$(C, K_1) \leftarrow_R \mathsf{KEnc}(\mathsf{pk})$	**if** $C_i \notin \mathcal{C}_{\mathbf{enc}}$ **and**
$\mathcal{C}_{\mathbf{enc}} := \emptyset$	$\mathcal{C}_{\mathbf{enc}} := \mathcal{C}_{\mathbf{enc}} \cup \{C\}$	**if** $\mathsf{pred}_i(K_i) = 1$
$b' \leftarrow_R \mathcal{A}^{\mathcal{O}_{\mathrm{enc}}, \mathcal{O}_{\mathrm{dec}}(\cdot,\cdot)}(\mathsf{pk})$	**return** (C, K_b)	**return** K_i
if $b = b'$ **return** 1		**else return** $\perp$
else return 0		

Here $\mathsf{pred}_i \colon \mathcal{K}(\lambda) \mapsto \{0,1\}$ denotes the predicate sent in the i-th decryption query, which is required to be provided as the description of a polynomial time algorithm (which can be enforced for instance by requiring it to be given in form of a circuit). Let further Q_{dec} be the number of total decryption queries made by $\mathcal{A}$ during the experiment, which are independent of the environment (hereby we refer to the environment the adversary runs in) without loss of generality. The uncertainty of knowledge about the keys corresponding to decryption queries is defined as

$$\mathrm{uncert}_{\mathcal{A}}(\lambda) := \frac{1}{Q_{\mathrm{dec}}} \sum_{i=1}^{Q_{\mathrm{dec}}} \Pr_{K \leftarrow_R \mathcal{K}(\lambda)}[\mathsf{pred}_i(K) = 1].$$

We say that the key encapsulation mechanism KEM is *IND-CCCA* secure, if for all PPT adversaries with negligible $\mathrm{uncert}_{\mathcal{A}}(\lambda)$, for the advantage we have

$$\mathrm{Adv}^{\mathrm{ccca}}_{\mathsf{KEM},\mathcal{A}}(\lambda) := \left| \Pr[\mathrm{Exp}^{\mathrm{ccca}}_{\mathsf{KEM},\mathcal{A}}(\lambda) = 1] - \frac{1}{2} \right| \leq \mathsf{negl}(\lambda).$$

$\underline{\mathsf{KGen}(1^\lambda)}$:
$(\mathsf{ppk}, \mathsf{psk}) \leftarrow_R \mathsf{PGen}(1^\lambda)$
$\mathbf{k}_0, \mathbf{k}_1 \leftarrow_R \mathbb{Z}_p^{2k}$
return
$\mathsf{pk} := (\mathsf{ppk}, [\mathbf{k}_0^\top \mathbf{A}], [\mathbf{k}_1^\top \mathbf{A}])$
$\mathsf{sk} := (\mathsf{psk}, \mathbf{k}_0, \mathbf{k}_1)$

$\underline{\mathsf{KEnc}(\mathsf{pk})}$:
$\mathbf{r} \leftarrow_R \mathbb{Z}_p^k$
$[\mathbf{t}] := [\mathbf{A}]\mathbf{r}$
$(\Pi, [\kappa]) := \mathsf{PPrv}(\mathsf{ppk}, [\mathbf{t}], \mathbf{r})$
$\tau := \mathsf{H}(\overline{[\mathbf{t}]})$
return
$C := ([\mathbf{t}], \Pi)$
$K := ([\mathbf{k}_0^\top \mathbf{A}] + \tau[\mathbf{k}_1^\top \mathbf{A}])\mathbf{r} + [\kappa]$

$\underline{\mathsf{KDec}(\mathsf{pk}, \mathsf{sk}, C)}$:
parse $C := ([\mathbf{t}], \Pi)$
$(b, [\kappa]) := \mathsf{PVer}(\mathsf{psk}, [\mathbf{t}], \Pi)$
if $b = 0$ **return** $\perp$
$\tau := \mathsf{H}(\overline{[\mathbf{t}]})$
return $K := (\mathbf{k}_0 + \tau\mathbf{k}_1)^\top [\mathbf{t}] + [\kappa]$

Figure 3.6: Construction of the KEM

Note that the term $\mathrm{uncert}_{\mathcal{A}}(\lambda)$ in the final reduction (proving IND-CCA security of the hybrid encryption scheme consisting of an unconditionally one-time secure authenticated encryption scheme and an IND-CCCA secure KEM) is statistically small (due to the fact that the symmetric building block is unconditionally secure). Thus we are able obtain a tight security reduction even if the term $\mathrm{uncert}_{\mathcal{A}}(\lambda)$ is multiplied by the number of encryption and decryption queries in the security loss (as it will be the case for our construction).

3.4 Our Tightly Secure Key Encapsulation Mechanism

In this section we present our CCCA-secure KEM that builds upon a qualified proof system for the OR-language as presented in Section 3.2.

Ingredients. Let $\mathsf{pars}_{\mathsf{PS}}$ be the public parameters for the underlying qualified proof system comprising $\mathcal{G} = (\mathbb{G}, p, P)$ and $\mathbf{A}, \mathbf{A}_0 \in \mathbb{Z}_p^{2k \times k}$. Recall that $\mathcal{L} = \mathrm{span}([\mathbf{A}])$, $\mathcal{L}^{\mathrm{snd}} = \mathrm{span}([\mathbf{A}]) \cup \mathrm{span}([\mathbf{A}_0])$ and $\mathcal{L}^{\mathrm{ext}} = \mathrm{span}([\mathbf{A}]) \cup \mathrm{span}([\mathbf{A}_0]) \cup \mathrm{span}([\mathbf{A}_1])$ (for $\mathbf{A}_1 \in \mathbb{Z}_p^{2k \times k}$). Let further $\mathcal{H}$ be a collosion resistant hash function generator returning functions of the form $\mathsf{H}\colon \mathbb{G}^k \to \{0,1\}^\lambda$ and let $\mathsf{H} \leftarrow_R \mathcal{H}$. We will sometimes interpret values $\tau \in \{0,1\}^\lambda$ in the image of H as elements in $\mathbb{Z}_p$ via the map $\tau \mapsto \sum_{i=1}^{\lambda} \tau_i \cdot 2^{i-1}$.

In the following we assume that all algorithms implicitly have access to the public parameters $\mathsf{pars}_{\mathsf{KEM}} := (\mathsf{pars}_{\mathsf{PS}}, \mathsf{H})$.

Proof systems. We employ an $\mathcal{L}^{\mathrm{snd}}$-qualified and $\mathcal{L}^{\mathrm{ext}}$-extensible proof system $\mathsf{PS} := (\mathsf{PGen}, \mathsf{PPrv}, \mathsf{PVer}, \mathsf{PSim})$ for the language $\mathcal{L}$ as provided in Fig. 3.1. We additionally require that the key space is a subset of $\mathbb{G}$, which is satisfied by our construction in Section 3.2.

Construction. The construction of the KEM is given in Fig. 3.6.

Efficiency. When using our qualified proof system from Section 3.2 to instantiate PS, the public parameters comprise $4k^2$ group elements (plus the descriptions of the

group itself and three hash functions). Further public keys and ciphertexts of our KEM contain $k^3 + k^2 + 4k$, resp. $k^2 + 2k$ group elements.

We stress that our scheme does not require pairings and can be implemented with $k = 1$, resulting in a tight security reduction to the DDH assumption in $\mathbb{G}$. As in this case the upper entries of the matrix $\mathbf{A}$ is 1, we get by with 3 group elements in the public parameters. Further, note that in case $k = 1$ public keys and ciphertexts contain 6, resp. 3 group elements. Compared to the GHKW scheme [GHKW16], our scheme thus has ciphertexts of the same size, but significantly smaller public keys.

Without any optimizations, encryption and decryption take $2k^3 + 5k^2 + 5k$, resp. $3k^3+4k^2+9k$ exponentiations ($2k^2$ for computing $[\mathbf{t}]$, $2k^3+3k^2+3k$ for computing π and $[\kappa]$ and $2k$ for computing K, resp. $3k^3 + 4k^2 + 5k$ for verifying π and computing $[\kappa]$ and $4k$ for computing K). For DDH this results in 11 resp. 16 exponentiations (in encryption one exponentiation can be saved due to the form of $\mathbf{A}$). Since most of these are multi-exponentiations, however, there is room for optimizations. In comparison, encryption and decyption in the GHKW scheme take $3k^2 + k$, resp. $3k$ exponentiations (plus about λk group operations for encryption, and again with room for optimizations). The main reason for our somewhat less efficient operations is the used qualified proof system. We explicitly leave open the construction of a more efficient proof system.

To turn the KEM into a IND-CCA secure hybrid encryption scheme, we require a quantitatively stronger security of the symmetric building block than [GHKW16]. Namely, the uncertainty $\mathrm{uncert}_{\mathcal{A}}(\lambda)$ in our scheme has a stronger dependency on the number of queries ($Q_{\mathrm{enc}} \cdot Q_{\mathrm{dec}}$ instead of $Q_{\mathrm{enc}} + Q_{\mathrm{dec}}$). This necessitates to increase the key size of the authenticated encryption scheme compared to [GHKW16]. Note though that one-time secure authenticated encryption schemes even exist unconditionally and therefore in the reduction proving security of the hybrid encryption scheme, the uncertainty $\mathrm{uncert}_{\mathcal{A}}(\lambda)$ will be statistically small.

Theorem 38 (Security of the KEM)**.** *If* PS *is* $\mathcal{L}^{\mathrm{snd}}$*-qualified and* $\mathcal{L}^{\mathrm{ext}}$*-extensible to* PS'*, if* H *is a collision resistant hash function and if the* $\mathcal{D}_{2k,k}$*-MDDH assumption holds in* $\mathbb{G}$*, then the key encapsulation mechanism* KEM *described in Fig. 3.6 is perfectly correct and IND-CCCA secure. More precisely, for every IND-CCCA adversary* $\mathcal{A}$ *that makes at most* Q_{enc} *encryption and* Q_{dec} *decryption queries, there exist adversaries* $\mathcal{B}^{\mathrm{mddh}}$*,* $\mathcal{B}^{\mathrm{snd}}$*,* $\mathcal{B}^{\mathrm{ind}}$*,* $\mathcal{B}^{\mathrm{snd}'}$ *and* $\mathcal{B}^{\mathrm{cr}}$ *with running time* $T(\mathcal{B}^{\mathrm{mddh}}) \approx T(\mathcal{B}^{\mathrm{snd}}) \approx T(\mathcal{B}^{\mathrm{ind}}) \approx T(\mathcal{B}^{\mathrm{snd}'}) \approx T(\mathcal{B}^{\mathrm{cr}}) \approx T(\mathcal{A}) + (Q_{\mathrm{enc}} + Q_{\mathrm{dec}}) \cdot \mathsf{poly}(\lambda)$ *respectively* $T(\mathcal{B}^{\mathrm{snd}'}) \approx T(\mathcal{A}) + (Q_{\mathrm{enc}} + Q_{\mathrm{enc}} \cdot Q_{\mathrm{dec}}) \cdot \mathsf{poly}(\lambda)$ *where* poly *is a polynomial independent of* $T(\mathcal{A})$*, and such that*

$$\begin{aligned}
\mathrm{Adv}^{\mathrm{ccca}}_{\mathsf{KEM},\mathcal{A}}(\lambda) \leq\ & \frac{1}{2} \cdot \mathrm{Adv}^{\mathrm{snd}}_{\mathcal{L}^{\mathrm{snd}},\mathsf{PS},\mathcal{B}^{\mathrm{snd}}}(\lambda) + \frac{1}{2} \cdot \mathrm{Adv}^{\mathrm{ind}}_{\mathcal{L}^{\mathrm{snd}},\mathsf{PS},\mathsf{PS}',\mathcal{B}^{\mathrm{ind}}}(\lambda) \\
& + (2\lambda + 2 + k) \cdot \mathrm{Adv}^{\mathrm{mddh}}_{\mathbb{G},\mathcal{D}_{2k,k},\mathcal{B}^{\mathrm{mddh}}}(\lambda) \\
& + \frac{\lambda}{2} \cdot \mathrm{Adv}^{\mathrm{snd}}_{\mathcal{L}^{\mathrm{ext}},\mathsf{PS}',\mathcal{B}^{\mathrm{snd}'}}(\lambda) \\
& + \frac{\lambda + 2}{2} \cdot Q_{\mathrm{enc}} \cdot Q_{\mathrm{dec}} \cdot \mathrm{uncert}_{\mathcal{A}}(\lambda) \\
& + \mathrm{Adv}^{\mathrm{cr}}_{\mathsf{H},\mathcal{B}^{\mathrm{cr}}}(\lambda) + Q_{\mathrm{enc}} \cdot 2^{-\Omega(\lambda)}.
\end{aligned}$$

Proof. We use a series of games to prove the claim. We denote the probability that

#	**ch. t**	**ch.** $[\kappa]$	$\mathcal{O}_{\text{dec}}$ **checks**	**remark**
$\mathbf{G}_0$	$\mathbf{A}$	PPrv		IND-CCCA
$\mathbf{G}_1$	$\mathbf{A}$	PPrv	τ fresh	coll. resist. of H
$\mathbf{G}_2$	$\mathbf{A}$	PSim	τ fresh	ZK of PS
$\mathbf{G}_3$	$\mathbf{A}_0$	PSim	τ fresh	$\mathcal{D}_{2k,k}$-MDDH
$\mathbf{G}_4$	$\mathbf{A}_0$	PSim	τ fresh, $[\mathbf{t}] \in \text{span}([\mathbf{A}])$	Lemma 39
$\mathbf{G}_5$	$\mathbf{A}_0$	rand	τ fresh, $[\mathbf{t}] \in \text{span}([\mathbf{A}])$	$\mathcal{D}_{2k,k}$-MDDH

Figure 3.7: Security of the KEM. Here column "**ch. t**" refers to the vector computed by $\mathcal{O}_{\text{enc}}$ as part of the challenge ciphertexts, where $\mathbf{A}$ indicates that $[\mathbf{t}] \leftarrow_R \text{span}([\mathbf{A}])$, for instance. Column "**ch.** $[\kappa]$" refers to the key computed by $\mathcal{O}_{\text{enc}}$ as part of the key K. In the column "$\mathcal{O}_{\text{dec}}$ **checks**" we describe what $\mathcal{O}_{\text{dec}}$ checks on input $C = (\mathsf{pred}, ([\mathbf{t}], \Pi))$ additionally to $C \notin \mathcal{C}_{\text{enc}}$ and $\text{pred}(K) = 1$. By a *fresh* tag $\tau := \mathsf{H}(\overline{[\mathbf{t}]})$ we denote a tag not previously used in any encryption query. In case the check fails, the decryption oracle outputs $\perp$.

the adversary $\mathcal{A}$ wins the i-th Game $\mathbf{G}_i$ by ε_i. An overview of all games is given in Fig. 3.7.

The goal is to randomize the keys of all challenge ciphertexts and thereby reducing the advantage of the adversary to 0. The methods employed here for a tight security reduction require us to ensure that $\mathcal{O}_{\text{dec}}$ aborts on ciphertexts which are not in the span of $[\mathbf{A}]$, as we will no longer be able to answer those. The justification of this step relies crucially on the additional consistency proof Π and is outsourced in Theorem 39.

Game $\mathbf{G}_0$: This game is the IND-CCCA security game (Theorem 37).

Transition $\mathbf{G}_0 \rightsquigarrow \mathbf{G}_1$: From game $\mathbf{G}_1$ on, we restrict the adversary to decryption queries with a fresh tag, that is, a tag which has not shown up in any previous encryption query. There are two conceivable bad events, where the adversary reuses a tag.

The first event is due to a collision of the hash function. That is, $\mathcal{A}$ provides a decryption query $([\mathbf{t}], \Pi)$, such that there exists a challenge ciphertext $[\mathbf{t}']$ from a previous encryption query with $\overline{[\mathbf{t}]} \neq \overline{[\mathbf{t}']}$, but $\mathsf{H}(\overline{[\mathbf{t}]}) = \mathsf{H}(\overline{[\mathbf{t}']})$. In that case we can straightforwardly employ $\mathcal{A}$ to obtain an adversary $\mathcal{B}$ attacking the collision resistance of H in time $T(\mathcal{B}) \approx T(\mathcal{A}) + (Q_{\text{enc}} + Q_{\text{dec}}) \cdot \mathsf{poly}(\lambda)$ for a polynomial poly independent of $T(\mathcal{A})$. Thereby we obtain an upper bound on the described event of $\text{Adv}^{\text{cr}}_{\mathsf{H},\mathcal{B}}(\lambda)$.

In the second event, $\mathcal{A}$ provides a valid decryption query $([\mathbf{t}], \Pi)$, such that $\overline{[\mathbf{t}]} = \overline{[\mathbf{t}']}$ for a previous challenge ciphertext $[\mathbf{t}'] \neq [\mathbf{t}]$. By the properties of PS, the proof corresponding to a ciphertext $[\mathbf{t}]$ is unique, which in particular implies $[\mathbf{t}] \notin \text{span}([\mathbf{A}])$. We bound the probability that $\mathcal{A}$ submits a valid

decryption query $([\mathbf{t}], \Pi)$ such that $[\mathbf{t}] \notin \mathrm{span}([\mathbf{A}])$ by $Q_{\mathrm{dec}} \cdot \mathrm{uncert}_{\mathcal{A}}(\lambda)$, using a series of hybrids: For $i = 0, \ldots, Q_{\mathrm{dec}}$ let $\mathbf{G}_{0.i}$ be defined like $\mathbf{G}_0$, except $\mathcal{O}_{\mathrm{dec}}$ checks the freshness of τ for the first i queries and operates as in game $\mathbf{G}_0$ from the $(i+1)$-st query on. Note that game $\mathbf{G}_{0.0}$ equals $\mathbf{G}_0$ and game $\mathbf{G}_{0.Q_{\mathrm{dec}}}$ equals $\mathbf{G}_1$. We show that for all $i \in \{0, \ldots, Q_{\mathrm{dec}} - 1\}$:

$$|\varepsilon_{0.i} - \varepsilon_{0.(i+1)}| \leq \Pr_{K \leftarrow_R \mathcal{K}}[\mathsf{pred}_{i+1}(K) = 1].$$

Game $\mathbf{G}_{0.i}$ and game $\mathbf{G}_{0.(i+1)}$ only differ when the $(i+1)$-st query to $\mathcal{O}_{\mathrm{dec}}$ is valid with $\overline{[\mathbf{t}]} = \overline{[\mathbf{t}']}$ for a previous challenge ciphertext $[\mathbf{t}'] \neq [\mathbf{t}]$. As all challenge ciphertexts are in $\mathrm{span}([\mathbf{A}])$, they do not reveal anything about $\mathbf{k}_0$ beyond the public key $[\mathbf{k}_0^\top \mathbf{A}]$. Thus, for $[\mathbf{t}] \notin \mathrm{span}([\mathbf{A}])$, the value $\mathbf{k}_0^\top [\mathbf{t}]$ looks uniformly random from the adversary's point of view, proving the claimed distance between game $\mathbf{G}_{0.i}$ and game $\mathbf{G}_{0.(i+1)}$. Altogether we obtain

$$|\varepsilon_0 - \varepsilon_1| \leq \mathrm{Adv}^{\mathrm{cr}}_{\mathsf{H},\mathcal{B}}(\lambda) + Q_{\mathrm{dec}} \cdot \mathrm{uncert}_{\mathcal{A}}(\lambda).$$

Transition $\mathbf{G}_1 \rightsquigarrow \mathbf{G}_2$: From $\mathbf{G}_2$ on, the way challenge ciphertexts are computed is changed. Namely, the simulation algorithmen $\mathsf{PSim}(\mathsf{psk}, [\mathbf{t}])$ is used instead of $\mathsf{PPrv}(\mathsf{ppk}, [\mathbf{t}], \mathbf{r})$ to compute $(\Pi, [\kappa])$. Since for all challenge ciphertexts we have $[\mathbf{t}] \in \mathcal{L}$, the proofs and keys are equal by the perfect zero-knowledge property of PS, and thus we have

$$\varepsilon_1 = \varepsilon_2.$$

Transition $\mathbf{G}_2 \rightsquigarrow \mathbf{G}_3$: Game $\mathbf{G}_3$ is like $\mathbf{G}_2$ except the vectors $[\mathbf{t}]$ in the challenge ciphertexts are chosen randomly in the span of $[\mathbf{A}_0]$.

We first employ the Q_{enc}-fold $\mathcal{D}_{2k,k}$-MDDH assumption to tightly switch the vectors in the challenge ciphertexts from $\mathrm{span}([\mathbf{A}])$ to uniformly random vectors over $\mathbb{G}^{2k}$. Next we use the Q_{enc}-fold $\mathcal{U}_{2k,k}$-MDDH assumption to switch these vectors from random to $[\mathbf{A}_0\mathbf{r}]$.

To be specific, we build adversaries $\mathcal{B}$, $\mathcal{B}'$ such that for a polynomial poly independent of $T(\mathcal{A})$ we have $T(\mathcal{B}) \approx T(\mathcal{B}') \approx T(\mathcal{A}) + (Q_{\mathrm{enc}} + Q_{\mathrm{dec}}) \cdot \mathsf{poly}(\lambda)$ and

$$|\varepsilon_2 - \varepsilon_3| \leq \mathrm{Adv}^{Q_{\mathrm{enc}}\text{-mddh}}_{\mathbb{G},\mathcal{D}_{2k,k},\mathcal{B}}(\lambda) + \mathrm{Adv}^{Q_{\mathrm{enc}}\text{-mddh}}_{\mathbb{G},\mathcal{U}_{2k,k},\mathcal{B}'}(\lambda).$$

Let $([\mathbf{A}], [\mathbf{v}_1| \ldots |\mathbf{v}_{Q_{\mathrm{enc}}}])$ with $[\mathbf{A}] \in \mathbb{G}^{2k \times k}$ and $[\mathbf{V}] := [\mathbf{v}_1| \ldots |\mathbf{v}_{Q_{\mathrm{enc}}}] \in \mathbb{G}^{2k \times Q_{\mathrm{enc}}}$ be the Q_{enc}-fold $\mathcal{D}_{2k,k}$-MDDH challenge received by $\mathcal{B}$. Then, the adversary $\mathcal{B}$ samples $(\mathsf{ppk}, \mathsf{psk}) \leftarrow_R \mathsf{PGen}(1^\lambda)$, $\mathbf{k}_0, \mathbf{k}_1 \leftarrow_R \mathbb{Z}_p^{2k}$, $b \leftarrow_R \{0,1\}$ and sends the public key $\mathsf{pk} := (\mathsf{ppk}, [\mathbf{k}_0^\top \mathbf{A}], [\mathbf{k}_1^\top \mathbf{A}])$ to $\mathcal{A}$.

On the i-th query to $\mathcal{O}_{\mathrm{enc}}$, $\mathcal{B}$ sets the challenge ciphertext to $[\mathbf{t}] := [\mathbf{v}_i]$, next computes $\tau := \mathsf{H}(\overline{[\mathbf{t}]})$, $(\Pi, [\kappa]) := \mathsf{PSim}(\mathsf{psk}, [\mathbf{v}_i])$ and finally $K_1 := (\mathbf{k}_0^\top + \tau \mathbf{k}_1^\top)[\mathbf{t}]$ (and $K_0 \leftarrow_R \mathcal{K}(\lambda)$ as usual). As $\mathcal{B}$ has generated the secret key itself, for decryption queries it can simply follow $\mathsf{KDec}(\mathsf{pk}, \mathsf{sk}, C)$.

In case $[\mathbf{V}] = [\mathbf{AR}]$, $\mathcal{B}$ perfectly simulates game $\mathbf{G}_2$. In case $[\mathbf{V}]$ is uniformly random over $\mathbb{G}^{2k \times Q_{\mathrm{enc}}}$, $\mathcal{B}$ simulates an intermediary game $\mathbf{H}$, where the challenge ciphertexts are chosen uniformly at random. Analogously we construct an adversary $\mathcal{B}'$ on the Q_{enc}-fold $\mathcal{U}_{2k,k}$-MDDH assumption, who simulates game

$\mathbf{H}$ if $[\mathbf{V}]$ is uniformly at random over $\mathbb{G}^{2k\times Q_{\text{enc}}}$, and game $\mathbf{G}_3$, if $[\mathbf{V}] = [\mathbf{A}_0\mathbf{R}]$. Altogether this proves the claim stated above.

Finally, from Theorem 6 (random self-reducibility of $\mathcal{U}_{2k,k}$-MDDH), Theorem 5 ($\mathcal{D}_{2k,k}$-MDDH $\Rightarrow$ $\mathcal{U}_{2k,k}$-MDDH), and Theorem 3 (random self-reducibility of $\mathcal{D}_{2k,k}$-MDDH), we obtain an adversary $\mathcal{B}''$ such that $T(\mathcal{B}'') \approx T(\mathcal{A}) + (Q_{\text{enc}} + Q_{\text{dec}}) \cdot \mathsf{poly}(\lambda)$ where poly is independent of $T(\mathcal{A})$ and

$$|\varepsilon_2 - \varepsilon_3| \leq (1+k) \cdot \mathrm{Adv}^{\mathrm{mddh}}_{\mathbb{G},\mathcal{D}_{2k,k},\mathcal{B}''}(\lambda) + \frac{2}{p-1}.$$

Transition $\mathbf{G}_3 \rightsquigarrow \mathbf{G}_4$: We now restrict the adversary to decryption queries with $[\mathbf{t}] \in \mathrm{span}([\mathbf{A}])$. For the justification we refer to Theorem 39 .

Transition $\mathbf{G}_4 \rightsquigarrow \mathbf{G}_5$: In game $\mathbf{G}_5$, we change the keys $[\kappa]$ computed by $\mathcal{O}_{\text{enc}}$ to random over $\mathbb{G}$. This is justified as follows.

Firstly, we can replace $\mathbf{k}_0$ by $\mathbf{k}_0 + \mathbf{A}^\perp\mathbf{u}$ with $\mathbf{u} \leftarrow_R \mathbb{Z}_p^k$ and $\mathbf{A}^\perp \in \mathrm{orth}(\mathbf{A})$, as those are identically distributed. Note that this change does neither affect the public key, nor the decryption queries, since for all $\mathbf{t} \in \mathrm{span}(\mathbf{A})$, $\mathbf{t}^\top(\mathbf{k}_0 + \mathbf{A}^\perp\mathbf{u}) = \mathbf{t}^\top\mathbf{k}_0$. Thus, the term $\mathbf{A}^\perp\mathbf{u}$ only shows up when $\mathcal{O}_{\text{enc}}$ computes the value $[(\mathbf{A}^\perp\mathbf{u})^\top\mathbf{A}_0\mathbf{r}]$ for $\mathbf{r} \leftarrow_R \mathbb{Z}_p^k$ as part of the key K_1 (the key that is not chosen at random by the security experiment).

Secondly, the distributions $(\mathbf{A}^\perp\mathbf{u})^\top\mathbf{A}_0$ and $\mathbf{v}^\top \leftarrow_R \mathbb{Z}_p^{1\times k}$ are $1 - 2^{-\Omega(\lambda)}$-close.

Altogether, we obtain that $\mathcal{O}_{\text{enc}}$, on its j-th query for each $j \in [Q_{\text{enc}}]$, can compute key K_1 for $\mathbf{r}_j \leftarrow_R \mathbb{Z}_p^k$, and $\mathbf{v} \leftarrow_R \mathbb{Z}_p^k$ as

$$K_1 := \left[(\mathbf{k}_0 + \tau\mathbf{k}_1)^\top \mathbf{A}_0\mathbf{r}_j\right] + \boxed{[\mathbf{v}^\top\mathbf{r}_j]} + [\kappa].$$

We then switch from $([\mathbf{r}_j], [\mathbf{v}^\top\mathbf{r}_j])$ to $([\mathbf{r}_j], [z_j])$, where z_j is a uniformly random value over $\mathbb{G}$, using the Q_{enc}-fold $\mathcal{U}_k$-MDDH assumption as follows. On input $([\mathbf{B}], [\mathbf{h}_1|\ldots|\mathbf{h}_{Q_{\text{enc}}}])$ with $\mathbf{B} \leftarrow_R \mathcal{U}_k$ (that is $\mathbf{B} \in \mathbb{Z}_p^{(k+1)\times k}$) and $\mathbf{h}_1, \ldots, \mathbf{h}_{Q_{\text{enc}}} \in \mathbb{Z}_p^{k+1}$, $\mathcal{B}$ samples $(\mathsf{ppk}, \mathsf{psk}) \leftarrow_R \mathsf{PGen}(1^\lambda)$, $\mathbf{k}_0, \mathbf{k}_1 \leftarrow_R \mathbb{Z}_p^{2k}$, $b \leftarrow_R \{0,1\}$ and sends the public key $\mathsf{pk} := (\mathsf{ppk}, [\mathbf{k}_0^\top\mathbf{A}], [\mathbf{k}_1^\top\mathbf{A}])$ to $\mathcal{A}$. In the following for all $j \in Q_{\text{enc}}$ let $\overline{[\mathbf{h}_j]} \in \mathbb{G}^k$ comprise the upper k entries and $\underline{[\mathbf{h}_j]} \in \mathbb{G}$ the $(k+1)$-st entry of $[\mathbf{h}_j]$ and similar for $[\mathbf{B}]$ let $\overline{[\mathbf{B}]} \in \mathbb{G}^{k\times k}$ be the upper square matrix of $[\mathbf{B}]$ and $\underline{[\mathbf{B}]} \in \mathbb{G}^{1\times k}$ comprise the last row.

On the j-th encryption query, $\mathcal{B}$ sets $[\mathbf{t}] := \mathbf{A}_0\overline{[\mathbf{h}_j]}$ (and thus $[\mathbf{r}_j] := \overline{[\mathbf{h}_j]}$) and computes the key as

$$K_1 := \left[(\mathbf{k}_0 + \tau\mathbf{k}_1)^\top \mathbf{t}\right] + \boxed{\underline{[\mathbf{h}_j]}} + [\kappa].$$

The adversary $\mathcal{B}$ can answer decryption queries as usual using $\mathbf{k}_0$, as decryption queries outside $\mathcal{L}$ are rejected.

Now if $([\mathbf{B}], [\mathbf{h}_1|\ldots|\mathbf{h}_{Q_{\text{enc}}}])$ was a real $\mathcal{U}_k$-MDDH challenge, we have $\mathbf{h}_j = \mathbf{B}\mathbf{s}_j$ for a $\mathbf{s}_j \leftarrow_R \mathbb{Z}_p^k$ and thus we have $\mathbf{r}_j = \overline{\mathbf{B}}\mathbf{s}_j$ and $\underline{[\mathbf{h}_j]} = \underline{[\mathbf{B}]}\mathbf{s}_j = \underline{[\mathbf{B}]}\overline{\mathbf{B}}^{-1}\mathbf{r}_j$. Note that the distribution of $\underline{[\mathbf{B}]}\overline{\mathbf{B}}^{-1}$ is statistically close to the distribution

of $\mathbf{v}^\top$ and therefore $\mathcal{B}$ simulates game $\mathbf{G}_4$. In case $\mathbf{h}_j$ was chosen uniformly at random from $\mathbb{Z}_p^{k+1}$, the adversary $\mathcal{B}$ simulates game $\mathbf{G}_5$ instead. In the end adversary $\mathcal{B}$ can thus forward the output of $\mathcal{A}$ to its own experiment.

Finally, Theorem 5, Theorem 6 and Theorem 7 yield the existence of an adversary $\mathcal{B}'$ such that $T(\mathcal{B}') \approx T(\mathcal{A}) + (Q_{\text{enc}} + Q_{\text{dec}}) \cdot \mathsf{poly}(\lambda)$ where poly is a polynomial independent of $T(\mathcal{A})$, and

$$|\varepsilon_4 - \varepsilon_5| \leq \mathrm{Adv}^{\text{mddh}}_{\mathbb{G},\mathcal{D}_{2k,k},\mathcal{B}'}(\lambda) + 2^{-\Omega(\lambda)}.$$

Game $\mathbf{G}_5$: In this game, the keys K_1 computed by $\mathcal{O}_{\text{enc}}$ are uniformly random, since the value $[\kappa]$ which shows up in $K_1 := [(\mathbf{k}_0 + \tau\mathbf{k}_1)^\top \mathbf{t}] + [\kappa]$ is uniformly random for each call to $\mathcal{O}_{\text{enc}}$. The same holds true for the keys K_0 which are chosen at random from $\mathcal{K}(\lambda)$ throughout all games. Therefore, the output of $\mathcal{O}_{\text{enc}}$ is now independent of the bit b chosen in $\mathrm{Exp}^{\text{ccca}}_{\mathsf{KEM},\mathcal{A}}(\lambda)$. This yields

$$\varepsilon_5 = 0.$$

□

Lemma 39. *The security games G_3 and G_4 defined for the proof of Theorem 38 (security of the KEM, see Figure 3.7) are computationally indistinguishable. More precisely, for every IND-CCCA adversary $\mathcal{A}$ that makes at most Q_{enc} encryption and Q_{dec} decryption queries, there exist adversaries $\mathcal{B}^{\text{snd}}$, $\mathcal{B}^{\text{ind}}$, $\mathcal{B}^{\text{mddh}}$ and $\mathcal{B}^{\text{snd}'}$ with running time $T(\mathcal{B}^{\text{snd}}) \approx T(\mathcal{B}^{\text{ind}}) \approx T(\mathcal{B}^{\text{mddh}}) \approx T(\mathcal{A}) + (Q_{\text{enc}} + Q_{\text{dec}}) \cdot \mathsf{poly}(\lambda)$ respectively $T(\mathcal{B}^{\text{snd}'}) \approx T(\mathcal{A}) + (Q_{\text{enc}} + Q_{\text{enc}} \cdot Q_{\text{dec}}) \cdot \mathsf{poly}(\lambda)$, where poly is a polynomial independent of $T(\mathcal{A})$, and such that*

$$\begin{aligned}\varepsilon_3 \leq \varepsilon_4 &+ \frac{1}{2} \cdot \mathrm{Adv}^{\text{snd}}_{\mathcal{L}^{\text{snd}},\mathsf{PS},\mathcal{B}^{\text{snd}}}(\lambda) + \frac{1}{2} \cdot \mathrm{Adv}^{\text{ind}}_{\mathcal{L}^{\text{snd}},\mathsf{PS},\mathsf{PS}',\mathcal{B}^{\text{ind}}}(\lambda) \\ &+ 2\lambda \cdot \mathrm{Adv}^{\text{mddh}}_{\mathbb{G},\mathcal{D}_{2k,k},\mathcal{B}^{\text{mddh}}}(\lambda) + \frac{\lambda}{2} \cdot \mathrm{Adv}^{\text{snd}}_{\mathcal{L}^{\text{ext}},\mathsf{PS}',\mathcal{B}^{\text{snd}'}}(\lambda) \\ &+ \frac{\lambda + 2}{2} \cdot Q_{\text{enc}} \cdot Q_{\text{dec}} \cdot \mathrm{uncert}_{\mathcal{A}}(\lambda) + Q_{\text{enc}} \cdot 2^{-\Omega(\lambda)}.\end{aligned}$$

Proof. From game $\mathbf{G}_4$ on, decryption queries outside the span of $[\mathbf{A}]$ will always be answered with $\perp$ independently of the corresponding proof Π.

Games $\mathbf{G}_3$ and $\mathbf{G}_4$ behave the same, as long as an adversary $\mathcal{A}$ does not manage to submit a decryption query $(\mathsf{pred}, ([\mathbf{t}], \Pi))$ with $[\mathbf{t}] \notin \mathrm{span}([\mathbf{A}])$, on which $\mathcal{O}_{\text{dec}}$ does not abort in $\mathbf{G}_3$.

In the following we will introduce probabilities conditioned on the bit b, which determines whether the encryption oracle returns uniformly random keys or real keys. Namely for $i \in \{3, 4\}$ and $\beta \in \{0, 1\}$ let $\varepsilon_{i|\beta}$ denote the probability that $\mathcal{A}$ wins game $\mathbf{G}_i$ under the condition that $b = \beta$ was drawn by the challenger. We prove that $\mathbf{G}_3$ and $\mathbf{G}_4$ are computationally indistinguishable, by a case analysis, depending on the bit b.

For $b = 0$: the encryption oracle $\mathcal{O}_{\text{enc}}$ of the experiment $\mathrm{Exp}^{\text{ind-ccca}}_{\mathsf{KEM},\mathcal{A}}(\lambda)$ returns keys chosen uniformly at random from $\mathcal{K}(\lambda)$, thus, all the adversary can information theoretically learn about $\mathbf{k}_0$ is $[\mathbf{k}_0^\top \mathbf{A}]$ from the public key. We can use the

remaining entropy from $\mathbf{k}_0$ to argue that the adversary $\mathcal{A}$ can only submits queries $(\mathsf{pred}, ([\mathbf{t}], \Pi))$ to $\mathcal{O}_{\mathrm{dec}}$, for which the corresponding key does not satisfies pred.

Namely, we replace $\mathbf{k}_0$ by $\mathbf{k}_0 + \mathbf{A}^{\perp}\mathbf{u}$ for $\mathbf{A}^{\perp} \in \mathrm{orth}(\mathbf{A})$, and $\mathbf{u} \leftarrow_R \mathbb{Z}_p^k$ as both are distributed identically. This change does not affect the public key, but for all $[\mathbf{t}] \notin \mathrm{span}([\mathbf{A}])$ we have: $[\mathbf{t}]^{\top}\mathbf{A}^{\perp} \neq \mathbf{0}$, and $[\mathbf{t}]^{\top}\mathbf{A}^{\perp}\mathbf{u}$ is uniformly random over $\mathbb{G}$. Therefore, the probability that the decryption oracle accepts a query $(\mathsf{pred}, ([\mathbf{t}], \Pi))$ with $[\mathbf{t}] \notin \mathrm{span}([\mathbf{A}])$, in $\mathbf{G}_3$ for $b = 0$, is bounded by $\Pr_{K \in \mathcal{K}}[\mathsf{pred}(K) = 1]$. Via a hybrid argument across all decryption queries, we obtain

$$|\varepsilon_{3|0} - \varepsilon_{4|0}| \leq Q_{\mathrm{dec}} \cdot \mathrm{uncert}_{\mathcal{A}}(\lambda).$$

For $b = 1$: In the following we will call a query *critical*, if it is of the form $(\mathsf{pred}, ([\mathbf{t}], \Pi))$ with $[\mathbf{t}] \notin \mathrm{span}([\mathbf{A}])$ and the decryption oracle does not abort in the respective game. Our goal is to bound the event of $\mathcal{A}$ submitting such a query. More precisely, we give the corresponding game $\mathbf{H}_0$ in Fig. 3.8, where $\mathcal{A}$ gets the public key pk as input and access to the oracles $\mathcal{O}_{\mathrm{enc}}$ and $\mathcal{O}_{\mathrm{dec}}$. $\mathcal{A}$ wins if the decryption oracle returns *critical query* at some point. Note that except for the altered winning condition, the oracles behave as in game $\mathbf{G}_3$ for $b = 1$. We denote the probability that the adversary $\mathcal{A}$ wins game $\mathbf{H}_{\mathbf{x}}$ by $\varepsilon_{\mathbf{H}.\mathbf{x}}$. Note that we have

$$|\varepsilon_{3|1} - \varepsilon_{4|1}| \leq \varepsilon_{\mathbf{H}.0}$$

and thus altogether we obtain

$$|\varepsilon_3 - \varepsilon_4| \leq \frac{1}{2} \cdot \left(\varepsilon_{\mathbf{H}.0} + Q_{\mathrm{dec}} \cdot \mathrm{uncert}_{\mathcal{A}}(\lambda)\right).$$

In the following we will bound $\varepsilon_{\mathbf{H}.0}$ via a sequence of games. We give an overview of the games in Fig. 3.9.

We will always assume that the freshness of τ is checked by the decryption oracle (and the query is answered with $\perp$ if it fails). In all games, an adversary wins if it manages to submit a critical query.

Transition $\mathbf{H}_0 \rightsquigarrow \mathbf{H}_1$: We will first reject decryption queries outside $\mathcal{L}^{\mathrm{snd}}$. We justify this employing the constrained soundness of PS. Let $\mathcal{A}$ be an adversary distinguishing between games $\mathbf{H}_0$ and $\mathbf{H}_1$, that is an adversary submitting a successful decryption query outside $\mathcal{L}^{\mathrm{snd}}$ in $\mathbf{H}_0$. Then we construct an adversary $\mathcal{B}$ breaking constrained $\mathcal{L}^{\mathrm{snd}}$-soundness of PS as follows.

On receiving the public key ppk of PS, the adversary $\mathcal{B}$ samples $\mathbf{k}_0, \mathbf{k}_1 \leftarrow_R \mathbb{Z}_p^{2k}$, and sends $\mathsf{pk} := (\mathsf{ppk}, [\mathbf{k}_0^{\top}\mathbf{A}], [\mathbf{k}_1^{\top}\mathbf{A}])$ to $\mathcal{A}$.

On an encryption query of $\mathcal{A}$, the adversary $\mathcal{B}$ can employ its simulation oracle $\mathcal{O}_{\mathrm{sim}}$ to obtain $([\mathbf{t}], \Pi, [\kappa])$ with $[\mathbf{t}] \in \mathrm{span}([\mathbf{A}_0])$. The adversary $\mathcal{B}$ now computes $\tau := \mathsf{H}(\overline{[\mathbf{t}]})$ and sets $C := ([\mathbf{t}], \Pi)$ and $K := (\mathbf{k}_0 + \tau\mathbf{k}_1)^{\top}[\mathbf{t}] + [\kappa]$. Finally $\mathcal{B}$ returns (C, K) to $\mathcal{A}$.

To answer $\mathcal{A}$'s queries to $\mathcal{O}_{\mathrm{dec}}$ of the form $(\mathsf{pred}, ([\mathbf{t}], \Pi))$, we distinguish the following cases, where we use that $\mathcal{B}$ has access to $\mathbf{A}$ and $\mathbf{A}_0$. In all cases $\mathcal{B}$ computes $\tau := \mathbf{H}(\overline{[\mathbf{t}]})$ and defines the predicate $\mathsf{pred}' \colon K \mapsto \mathsf{pred}((\mathbf{k}_0 + \tau\mathbf{k}_1)^{\top}[\mathbf{t}] + K)$. Next $\mathcal{B}$ queries $\mathcal{O}_{\mathrm{ver}}$ on $([\mathbf{t}], \Pi, \mathsf{pred}')$.

$\underline{\mathrm{Exp}^{\mathbf{H}_x}_{\mathsf{KEM},\mathcal{A}}(\lambda)}$:
$(pk, sk) \leftarrow_R \mathsf{KGen}(1^\lambda)$
$\mathbf{v} \leftarrow_R \mathbb{Z}_p^{2k}$
$\mathcal{C}_{\mathbf{enc}} := \emptyset$
$\mathcal{A}^{\mathcal{O}_{\mathrm{enc}},\mathcal{O}_{\mathrm{dec}}(\cdot,\cdot)}(\mathsf{pk})$
if $\mathcal{O}_{\mathrm{dec}}$ **returned** *critical query*
  return 1
else return 0

$\underline{\mathcal{O}_{\mathrm{enc}}}$:
$\mathbf{r} \leftarrow_R \mathbb{Z}_p^k$
$[\mathbf{t}] := [\mathbf{A}_0]\mathbf{r}$
$\tau := \mathsf{H}([\mathbf{t}])$
$(\Pi, [\kappa]) := \mathsf{PSim}(\mathsf{ppk}, \mathsf{psk}, [\mathbf{t}])$
$C := ([\mathbf{t}], \Pi)$
$K := \left(\mathbf{k}_0 + \tau\mathbf{k}_1 + \mathbf{v}\right)^\top [\mathbf{t}] + [\kappa]$
$\mathcal{C}_{\mathrm{enc}} := \mathcal{C}_{\mathrm{enc}} \cup \{C\}$
return (C, K)

$\underline{\mathcal{O}_{\mathrm{dec}}(\mathsf{pred}, ([\mathbf{t}], \Pi))}$:
$(v, [\kappa]) := \mathsf{PVer}(\mathsf{psk}, [\mathbf{t}], \Pi)$
$\tau := \mathsf{H}([\mathbf{t}])$
if $([\mathbf{t}], \Pi) \notin \mathcal{C}_{\mathrm{enc}}$ **and** $v = 1$ **and** τ is fresh
  if $[\mathbf{t}] \in \mathrm{span}([\mathbf{A}])$
    $K := (\mathbf{k}_0 + \tau\mathbf{k}_1)^\top [\mathbf{t}] + [\kappa]$
    if $\mathsf{pred}(K) = 1$
      return K
  else **if** $[\mathbf{t}] \in \mathrm{span}([\mathbf{A}_0])$
    $K := \left(\mathbf{k}_0 + \tau\mathbf{k}_1 + \mathbf{v}\right)^\top [\mathbf{t}] + [\kappa]$
    if $\mathsf{pred}(K) = 1$
      return *critical query* **and abort**
return $\perp$

Figure 3.8: Games $\mathbf{H}_0$, $\boxed{\mathbf{H}_1}$ and $\mathbf{H}_2$

In case $[\mathbf{t}] \in \mathrm{span}([\mathbf{A}])$, the oracle returns either $\perp$ or a key $[\kappa]$ to $\mathcal{B}$. In the former case $\mathcal{B}$ forwards $\perp$ to $\mathcal{A}$, in the latter the key $K := (\mathbf{k}_0 + \tau\mathbf{k}_1)^\top [\mathbf{t}] + [\kappa]$.

If $[\mathbf{t}] \in \mathrm{span}([\mathbf{A}_0])$, the oracle $\mathcal{O}_{\mathrm{ver}}$ returns either $\perp$ or the adversary $\mathcal{B}$ has lost the constrained soundness game. In the former case, $\mathcal{B}$ forwards $\perp$ to $\mathcal{A}$. In the latter case the adversary $\mathcal{A}$ managed to submit a critical query in both games $\mathbf{H}_0$ and $\mathbf{H}_1$ and thus did not succeed in distinguishing between the two.

Finally, if $[\mathbf{t}] \notin \mathrm{span}([\mathbf{A}]) \cup \mathrm{span}([\mathbf{A}_0])$, the oracle $\mathcal{O}_{\mathrm{ver}}$ returns either $\perp$ (in which case $\mathcal{B}$ sends $\perp$ to $\mathcal{A}$), or the adversary $\mathcal{B}$ has win the constrained soundness game. Only in the last case does $\mathcal{A}$ distinguish between $\mathbf{H}_0$ and $\mathbf{H}_1$.

Altogether we obtain an adversary $\mathcal{B}$ breaking the constrained $\mathcal{L}^{\mathrm{snd}}$-soundness of PS in time $T(\mathcal{B}) \approx T(\mathcal{A}) + (Q_{\mathrm{enc}} + Q_{\mathrm{dec}}) \cdot \mathsf{poly}(\lambda)$, where poly is a polynomial

#	**proof system**	**ch.** $\mathbf{k}_\Delta^{\mathrm{enc}}(\tau)$	$\mathbf{k}_\Delta^{\mathrm{dec}}(\tau,[\mathbf{t}])$ **used by** $\mathcal{O}_{\mathrm{dec}}$ **on** $[\mathbf{c}]$ **for which** $[\mathbf{c}]^\top \mathbf{A}^\perp \neq [0]$	$\mathcal{O}_{\mathrm{dec}}$ **checks**	**game knows**	**remark**
$\mathbf{H}_0$	PS	0	0		$\mathbf{A}$	
$\mathbf{H}_1$	PS	0	0	$[\mathbf{t}] \in \mathcal{L}^{\mathrm{snd}}$	$\mathbf{A}$, $\mathbf{A}_0, \mathbf{A}_1$	$\mathcal{L}^{\mathrm{snd}}$-soundness
$\mathbf{H}_2$	PS	$\mathbf{v}$	$\mathbf{v}$	$[\mathbf{t}] \in \mathcal{L}^{\mathrm{snd}}$	$\mathbf{A}, \mathbf{A}_0, \mathbf{A}_1$	statistical
$\mathbf{H}_3$	PS′	$\mathbf{v}$	$\mathbf{v}$	$[\mathbf{t}] \in \mathcal{L}^{\mathrm{snd}}$	$\mathbf{A}, \mathbf{A}_0, \mathbf{A}_1$	$\mathcal{L}^{\mathrm{ext}}$-extensibility
$\mathbf{H}_4$	PS′	$\mathbf{v}$	$\mathbf{v}$		$\mathbf{A}$	win. chances increase
$\mathbf{H}_5$	PS′	$\mathbf{F}(\tau)$	$\{\mathbf{F}(\tau^{(j)})\}$		$\mathbf{A}$	see Figure 3.11

Figure 3.9: Security of the KEM. Column "**proof system**" describes the underlying proof system used, where PS′ is a $\mathcal{L}^{\mathrm{ext}}$-qualified proof system, such that PS and PS′ are $\mathcal{L}^{\mathrm{snd}}$-indistinguishable. Column "**ch.** $\mathbf{k}_\Delta^{\mathrm{enc}}(\tau)$" refers to the vector $\mathbf{k}_\Delta^{\mathrm{enc}}(\tau)$ used by $\mathcal{O}_{enc}$ when computing the key $K := [(\mathbf{k}_0 + \tau\mathbf{k}_1 + \mathbf{k}_\Delta^{\mathrm{enc}}(\tau))^\top \mathbf{t}] + [\kappa]$ for challenge ciphertexts. $\mathbf{v}$ denotes a value in $\mathbb{Z}_p^{2k}$ chosen uniformly random, $\mathbf{F}\colon \{0,1\}^\lambda \to \mathbb{Z}_p^{2k}$ denotes a random function and $\tau \coloneqq \mathsf{H}(\overline{[\mathbf{t}]})$. In the next column, we describe $\mathbf{k}_\Delta^{\mathrm{dec}}(\tau,[\mathbf{t}])$ used by $\mathcal{O}_{dec}$ when computing the set of valid keys $\mathcal{S}_K := \left\{ \left(\mathbf{k}_0 + \tau\mathbf{k}_1 + \mathbf{k}_\Delta^{\mathrm{dec}}(\tau^{(j)},[\mathbf{t}])\right)^\top [\mathbf{t}] + [\kappa] \,\middle|\, \tau^{(j)} \in \mathcal{Q}_{\mathrm{dec}} \right\}$ on queries containing $[\mathbf{t}]$ such that $\mathbf{t}^\top \mathbf{A}^\perp \neq 0$. Here $\tau^{(j)} \in \mathcal{Q}_{\mathrm{dec}}$ for $j \in \{1,\ldots,Q_{\mathrm{enc}}\}$ denotes the tag from the j-th encryption query. By the set notation we want to imply that the decryption oracle accepts a predicate if it evaluates to 1 on any key in $\mathcal{S}_K$. The column "$\mathcal{O}_{\mathrm{dec}}$ **checks**" refers to additional checks performed on decryption queries ahead of decryption. We always assume $\mathcal{O}_{\mathrm{dec}}$ checks the freshness of τ and therefore not list it explicitely in the table. In case any of the checks fails, $\mathcal{O}_{\mathrm{dec}}$ returns $\perp$. The column "**game knows**" refers to what the game must know with respect to $\mathbf{A}$, $\mathbf{A}_0$ and $\mathbf{A}_1$.

independent of $T(\mathcal{A})$, such that

$$|\varepsilon_{\mathbf{H}.0} - \varepsilon_{\mathbf{H}.1}| \leq \mathrm{Adv}^{\mathrm{snd}}_{\mathcal{L}^{\mathrm{snd}},\mathsf{PS},\mathcal{B}}(\lambda).$$

Transition $\mathbf{H}_1 \rightsquigarrow \mathbf{H}_2$: We alter the oracles in game $\mathbf{H}_2$ as described in Fig. 3.8, where the same $\mathbf{v} \leftarrow_R \mathbb{Z}_p^{2k}$ is used across all oracle calls. The appearance of the extra random term $\mathbf{v}$ in encryption and decryption queries with $[\mathbf{t}] \in \mathrm{span}([\mathbf{A}_0])$ is justified as follows. In an intermediary game we first replace $\mathbf{k}_0$ by $\mathbf{k}_0 + \mathbf{A}^\perp\mathbf{u}$, where $\mathbf{A}^\perp \in \mathrm{orth}(\mathbf{A})$ and $\mathbf{u} \leftarrow_R \mathbb{Z}_p^k$. This transition does not change the view of the adversaries as the keys $\mathbf{k}_0$ and $\mathbf{k}_0 + \mathbf{A}^\perp\mathbf{u}$ are both distributed uniformly random over $\mathbb{Z}_p^{2k}$.

Note that this change neither affects the public key, nor the keys computed by $\mathcal{O}_{\mathrm{dec}}$ when queried on inputs containing $[\mathbf{t}] \in \mathrm{span}([\mathbf{A}])$, since $(\mathbf{k}_0+\mathbf{A}^\perp\mathbf{u})^\top[\mathbf{t}] = \mathbf{k}_0^\top[\mathbf{t}]$.

Next for $\mathbf{A}_0^\perp \in \mathrm{orth}(\mathbf{A}_0)$ and $\mathbf{u}_0 \leftarrow_R \mathbb{Z}_p^k$ we replace $\mathbf{k}_0 + \mathbf{A}^\perp\mathbf{u}$ by $\mathbf{k}_0 + \mathbf{A}^\perp\mathbf{u} + \mathbf{A}_0^\perp\mathbf{u}_0$ in all encryption queries and decryption queries with $[\mathbf{t}] \in \mathrm{span}([\mathbf{A}_0])$, which does not change the adversary's view, since we have $(\mathbf{A}^\perp\mathbf{u})^\top[\mathbf{t}] = (\mathbf{A}^\perp\mathbf{u} + \mathbf{A}_0^\perp\mathbf{u}_0)^\top[\mathbf{t}]$.

With probability $1 - 2^{-\Omega(\lambda)}$ over the choices of $\mathbf{A}, \mathbf{A}_0$ the column vectors of $\mathbf{A}^\perp$ and $\mathbf{A}_0^\perp$ together form a basis of $\mathbb{Z}_p^{2k}$, and thus $\mathbf{A}^\perp\mathbf{u}+\mathbf{A}_0^\perp\mathbf{u}_0$ is distributed uniformly random over $\mathbb{Z}_p^{2k}$ with overwhelming probability and can be replaced by $\mathbf{v} \leftarrow \mathbb{Z}_p^{2k}$.

This yields

$$|\varepsilon_{\mathbf{H}.1} - \varepsilon_{\mathbf{H}.2}| \leq 2^{-\Omega(\lambda)}.$$

Transition $\mathbf{H}_2 \rightsquigarrow \mathbf{H}_3$: By the $\mathcal{L}^{\mathrm{ext}}$-extensibility of PS, there exists a proof system PS', such that PS and PS' are $\mathcal{L}^{\mathrm{snd}}$-indistinguishable. From game $\mathbf{H}_3$ on, we replace PS by PS'.

From an adversary $\mathcal{A}$ distinguishing between those to games, we can construct an adversary $\mathcal{B}$ breaking the $\mathcal{L}^{\mathrm{snd}}$-indistinguishability as follows, where $\mathcal{B}$ has either access to the oracles $\mathcal{O}^0_{\mathrm{sim}}$ and $\mathcal{O}^0_{\mathrm{ver}}$ of PS, or to the oracles $\mathcal{O}^1_{\mathrm{sim}}$ and $\mathcal{O}^1_{\mathrm{ver}}$ of PS' and has to distinguish between the two cases.

Note that we do not change the distribution of $[\mathbf{t}]$ in simulation queries in this step, that is in both games $[\mathbf{t}]$ is chosen uniformly at random from $\mathrm{span}([\mathbf{A}_0])$.

On receiving the public key ppk of PS, the adversary $\mathcal{B}$ samples $\mathbf{k}_0, \mathbf{k}_1 \leftarrow_R \mathbb{Z}_p^{2k}$, and sends $\mathsf{pk} := (\mathsf{ppk}, [\mathbf{k}_0^\top\mathbf{A}], [\mathbf{k}_1^\top\mathbf{A}])$ to $\mathcal{A}$. Now $\mathcal{B}$ can employ its simulation oracle $\mathcal{O}^\beta_{\mathrm{sim}}$ to answer decryption queries.

To answer $\mathcal{A}$'s queries to $\mathcal{O}_{\mathrm{dec}}$ of the form $(\mathsf{pred}, ([\mathbf{t}], \Pi))$, we distinguish the following cases, where we use that $\mathcal{B}$ has access to $\mathbf{A}$ and $\mathbf{A}_0$. All queries outside of $\mathcal{L}^{\mathrm{snd}}$ to the decryption oracle are answered with $\perp$ by $\mathcal{B}$. In case $[\mathbf{t}] \in \mathcal{L}^{\mathrm{snd}}$ the adversary $\mathcal{B}$ computes $\tau := \mathsf{H}(\overline{[\mathbf{t}]})$ and defines the predicate $\mathsf{pred}' \colon K \mapsto \mathsf{pred}((\mathbf{k}_0 + \tau\mathbf{k}_1)^\top[\mathbf{t}] + K)$. Next $\mathcal{B}$ queries $\mathcal{O}^\beta_{\mathrm{ver}}$ on $([\mathbf{t}], \Pi, \mathsf{pred}')$, to get either a key $[\kappa]$, or $\perp$. In the former case, $\mathcal{B}$ checks if $[\mathbf{t}] \in \mathrm{span}([\mathbf{A}])$, if this is the case, it returns the key $K := (\mathbf{k}_0 + \tau\mathbf{k}_1)^\top[\mathbf{t}] + [\kappa]$ to $\mathcal{A}$, if this is

$\underline{\mathcal{O}_{\text{enc}}}$:
$\mathbf{r} \leftarrow_R \mathbb{Z}_p^k$
$[\mathbf{t}] := [\mathbf{A}_0]\mathbf{r}$
$\tau := \mathsf{H}(\overline{[\mathbf{t}]})$
$(\Pi, [\kappa]) := \mathsf{PSim}'(\mathsf{ppk}, \mathsf{psk}, [\mathbf{t}])$
$C := ([\mathbf{t}], \Pi)$
$K := \left(\mathbf{k}_0 + \tau\mathbf{k}_1 + \mathbf{F}_i(\tau_{|i})\right)^\top [\mathbf{t}] + [\kappa]$
$\mathcal{C}_{\text{enc}} := \mathcal{C}_{\text{enc}} \cup \{C\}$
return (C, K)

$\underline{\mathcal{O}_{\text{dec}}(\mathsf{pred}, ([\mathbf{t}], \Pi))}$:
$(v, [\kappa]) := \mathsf{PVer}'(\mathsf{psk}, [\mathbf{t}], \Pi)$
$\tau^\star := \mathsf{H}(\overline{[\mathbf{t}]})$
if $([\mathbf{t}], \Pi) \notin \mathcal{C}_{\text{enc}}$ **and** $v = 1$ **and** τ is fresh
  if $[\mathbf{t}] \in \text{span}([\mathbf{A}])$
    $K := (\mathbf{k}_0 + \tau^\star\mathbf{k}_1)^\top [\mathbf{t}] + [\kappa]$
    if $\mathsf{pred}(K) = 1$
      return K
  else
    $\mathcal{S}_K := \left\{ \left(\mathbf{k}_0 + \tau^\star\mathbf{k}_1 + \mathbf{F}_i(\tau_{|i})\right)^\top [\mathbf{t}] + [\kappa] \;\middle|\; \tau \in \mathcal{Q}_{\text{enc}} \right\}$
    if $\exists K \in \mathcal{S}_K$ **such that** $\mathsf{pred}(K) = 1$
      return *critical query* **and abort**
return $\perp$

Figure 3.10: Oracles in Game $\mathbf{H}_{4.i.0}$

not the case it returns *critical query*, and ends the game. In the latter case, $\mathcal{B}$ sends $\perp$ to $\mathcal{A}$.

The adversary $\mathcal{B}$ now simulates game $\mathbf{H}_2$ in case $\beta = 0$ and game $\mathbf{H}_3$ in case $\beta = 1$, thus $\mathcal{B}$ can forward the output of $\mathcal{A}$ to its experiment.

Altogether we obtain thus an adversary $\mathcal{B}$ breaking the $\mathcal{L}^{\text{snd}}$-indistinguishability of PS and PS' in time $T(\mathcal{B}) \approx T(\mathcal{A}) + (Q_{\text{enc}} + Q_{\text{dec}}) \cdot \mathsf{poly}(\lambda)$, where poly is a polynomial independent of $T(\mathcal{A})$, such that

$$|\varepsilon_{\mathbf{H}.2} - \varepsilon_{\mathbf{H}.3}| \leq \text{Adv}^{\text{PS-ind}}_{\mathcal{L}^{\text{snd}},\mathsf{PS},\mathsf{PS}',\mathcal{B}}(\lambda),$$

Transition $\mathbf{H}_3 \rightsquigarrow \mathbf{H}_4$: From game $\mathbf{H}_4$ on, we again allow decryption queries outside $\mathcal{L}^{\text{snd}}$. This can only increase the winning chances of the adversary, as it does not change the view on non-critical queries. We thus have $\varepsilon_{\mathbf{H}.3} \leq \varepsilon_{\mathbf{H}.4}$.

Transition $\mathbf{H}_4 \rightsquigarrow \mathbf{H}_5$: To justify the transition from game $\mathbf{H}_4$ to game $\mathbf{H}_5$ we employ a hybrid argument comprising a number of games. We give an overview of these games in Fig. 3.11 and prove the reduction in the following.

Game $\mathbf{H}_{4.i.0}$: For $i = 0, \ldots, \lambda$, in $\mathbf{H}_{4.i.0}$ the adversary has access to the oracles $\mathcal{O}_{\text{enc}}$ and $\mathcal{O}_{\text{dec}}$ defined as described in Fig. 3.10, where by $\mathbf{F}_i : \{0,1\}^i \to \mathbb{Z}_p^{2k}$ we denote a random function applied to the first i bits $\tau_{|i}$ of τ.

#	ch. $[\mathbf{t}]$	ch. $\mathbf{k}_\Delta^{\mathrm{enc}}(\tau)$	$\mathbf{k}_\Delta^{\mathrm{dec}}(\tau,[\mathbf{t}])$ used by $\mathcal{O}_{\mathrm{dec}}$ on $[\mathbf{c}]$ for which $[\mathbf{c}]^\top\mathbf{A}^\perp \neq [0]$	$\mathcal{O}_{\mathrm{dec}}$ checks	game knows	remark
$\mathbf{H}_{4.i.0}$	$[\mathbf{A}_0]$	$\mathbf{F}_i(\tau_{\vert i})$	$\left\{\mathbf{F}_i(\tau_{\vert i}^{(j)})\right\}$		$\mathbf{A}$	$\mathbf{H}_{4.0.0} = \mathbf{H}_4$
$\mathbf{H}_{4.i.1}$	$[\mathbf{A}_{\tau_{i+1}}]$	$\mathbf{F}_i(\tau_{\vert i})$	$\left\{\mathbf{F}_i(\tau_{\vert i}^{(j)})\right\}$		$\mathbf{A}$	$\mathcal{D}_{2k,k}$-MDDH
$\mathbf{H}_{4.i.2}$	$[\mathbf{A}_{\tau_{i+1}}]$	$\mathbf{F}_i(\tau_{\vert i})$	$\left\{\mathbf{F}_i(\tau_{\vert i}^{(j)})\right\}$	$[\mathbf{t}] \in \mathcal{L}^{\mathrm{ext}}$	$\mathbf{A}, \mathbf{A}_0, \mathbf{A}_1$	$\mathcal{L}^{\mathrm{ext}}$-soundness
$\mathbf{H}_{4.i.3}$	$[\mathbf{A}_{\tau_{i+1}}]$	$\underline{\tau_{i+1}=0:}$ $\mathbf{A}_0^\perp\widetilde{\mathbf{F}}_i^{(0)}(\tau_{\vert i}) + \mathbf{A}_1^\perp\mathbf{F}_i^{(1)}(\tau_{\vert i})$ $\underline{\tau_{i+1}=1:}$ $\mathbf{A}_0^\perp\mathbf{F}_i^{(0)}(\tau_{\vert i}) + \mathbf{A}_1^\perp\widetilde{\mathbf{F}}_i^{(1)}(\tau_{\vert i})$	$\underline{\text{if } [\mathbf{t}] \in \mathrm{span}([\mathbf{A}_0]):}$ $\left\{\mathbf{A}_0^\perp\widetilde{\mathbf{F}}_i^{(0)}(\tau_{\vert i}^{(j)}) + \mathbf{A}_1^\perp\mathbf{F}_i^{(1)}(\tau_{\vert i}^{(j)})\right\}$ $\underline{\text{if } [\mathbf{t}] \in \mathrm{span}([\mathbf{A}_1]):}$ $\left\{\mathbf{A}_0^\perp\mathbf{F}_i^{(0)}(\tau_{\vert i}^{(j)}) + \mathbf{A}_1^\perp\widetilde{\mathbf{F}}_i^{(1)}(\tau_{\vert i}^{(j)})\right\}$	$[\mathbf{t}] \in \mathcal{L}^{\mathrm{ext}}$	$\mathbf{A}, \mathbf{A}_0, \mathbf{A}_1$	change of basis
$\mathbf{H}_{4.i.4}$	$[\mathbf{A}_{\tau_{i+1}}]$	$\mathbf{F}_{i+1}(\tau_{\vert i+1})$	$\left\{\mathbf{F}_{i+1}(\tau_{\vert i}^{(j)} d_{[\mathbf{t}]})\right\}$	$[\mathbf{t}] \in \mathcal{L}^{\mathrm{ext}}$	$\mathbf{A}, \mathbf{A}_0, \mathbf{A}_1$	conceptual
$\mathbf{H}_{4.i.5}$	$[\mathbf{A}_{\tau_{i+1}}]$	$\mathbf{F}_{i+1}(\tau_{\vert i+1})$	$\left\{\mathbf{F}_{i+1}(\tau_{\vert i}^{(j)} d_{[\mathbf{t}]})\right\}$		$\mathbf{A}, \mathbf{A}_0, \mathbf{A}_1$	win. chances increase
$\mathbf{H}_{4.i.6}$	$[\mathbf{A}_{\tau_{i+1}}]$	$\mathbf{F}_{i+1}(\tau_{\vert i+1})$	$\left\{\mathbf{F}_{i+1}(\tau_{\vert i}^{(j)} b), b \in \{0,1\}\right\}$		$\mathbf{A}$	win. chances increase
$\mathbf{H}_{4.i.7}$	$[\mathbf{A}_{\tau_{i+1}}]$	$\mathbf{F}_{i+1}(\tau_{\vert i+1})$	$\left\{\mathbf{F}_{i+1}(\tau_{\vert i+1}^{(j)})\right\}$		$\mathbf{A}$	$\mathbf{F}$ hard to guess on non-queried values

Figure 3.11: Hybrid Games for Randomization. Columns are almost according to Figure 3.9. Additionally column "**ch.** $[\mathbf{t}]$" refers to the vector computed by $\mathcal{O}_{enc}$ as part of the challenge ciphertexts, where $\mathbf{A}$ indicates that $\mathbf{t} \leftarrow_R \mathrm{span}(\mathbf{A})$, for instance. For $i = 0, \ldots, \lambda$ by $\mathbf{F}_i : \{0,1\}^i \to \mathbb{Z}_p^{2k}$ and further by $\mathbf{F}_i^{(0)}, \mathbf{F}_i^{(1)}, \widetilde{\mathbf{F}}_i^{(0)}, \widetilde{\mathbf{F}}_i^{(1)} : \{0,1\}^i \to \mathbb{Z}_p^k$ we denote random functions, such that for all $\rho \in \{0,1\}^i$ and for a choice $\mathbf{A}_0^\perp \in \mathrm{orth}([\mathbf{A}_0])$ and $\mathbf{A}_1^\perp \in \mathrm{orth}([\mathbf{A}_1])$ we have $\mathbf{F}_i(\rho) = \mathbf{A}_0^\perp\mathbf{F}_i^{(0)}(\rho) + \mathbf{A}_1^\perp\mathbf{F}_i^{(1)}(\rho)$. Apart from this relation we require the functions to be independent. We set $d_{[\mathbf{t}]} = 0$ if $[\mathbf{t}] \in \mathrm{span}([\mathbf{A}_0])$ and $d_{[\mathbf{t}]} = 1$ if $[\mathbf{t}] \in \mathrm{span}([\mathbf{A}_1])$. We always assume $\mathcal{O}_{\mathrm{dec}}$ checks the freshness of τ and therefore do not list it explicitly in the table. In case any of the checks fails, $\mathcal{O}_{\mathrm{dec}}$ returns $\perp$.

Note that in previous games ($\mathbf{H}_0$ to $\mathbf{H}_4$), for a statement $[\mathbf{t}] \notin \mathrm{span}([\mathbf{A}])$, $\mathcal{O}_{\mathrm{dec}}(\mathsf{pred}, ([\mathbf{t}], \Pi))$ computes one key K when the proof Π is valid, and return this key if $\mathsf{pred}(K) = 1$.

In game $\mathbf{H}_{4.i.0}$, instead, the decryption oracle will accept a query $(\mathsf{pred}, ([\mathbf{t}], \Pi))$ outside $\mathrm{span}([\mathbf{A}])$ as critical, if additionally to a valid proof Π, the corresponding predicate pred evaluates to 1 on any of the keys in the set

$$\mathcal{S}_K := \left\{ \left[\left(\mathbf{k}_0 + \tau^\star \mathbf{k}_1 + \mathbf{F}_i(\tau_{|i})\right)^\top \mathbf{t} \right] + [\kappa] \,\middle|\, \tau \in \mathcal{Q}_{\mathrm{enc}} \right\},$$

where $\tau^\star := \mathsf{H}(\overline{[\mathbf{t}]})$ and $\mathcal{Q}_{\mathrm{enc}}$ denotes the set of tags previously computed by $\mathcal{O}_{\mathrm{enc}}$. As for $i = 0$ the function $\mathbf{F}_i = \mathbf{F}_0$ is a constant random value in $\mathbb{Z}_p^{2k}$, independent from its input τ, we have $\mathbf{H}_{4.0.0} = \mathbf{H}_4$. Also note that $\mathbf{H}_{4.\lambda.0} = \mathbf{H}_5$.

Transition $\mathbf{H}_{4.i.0} \rightsquigarrow \mathbf{H}_{4.i.1}$: For $i = 0, \ldots, \lambda - 1$, $\mathbf{H}_{4.i.1}$ is defined as $\mathbf{H}_{4.i.0}$ except $\mathcal{O}_{\mathrm{enc}}$ computes ciphertexts of the form $[\mathbf{t}] := [\mathbf{A}_{\tau_{i+1}}\mathbf{r}]$, where τ_{i+1} denotes the $(i+1)$-st bit of τ, instead of $[\mathbf{A}_0\mathbf{r}]$ in $\mathbf{H}_{4.i.0}$. We justify this transition by applying the $\mathcal{U}_{2k,k}$-MDDH assumption twice. First we use it once with respect to $[\mathbf{A}_0]$ to tightly switch vectors from $[\mathbf{A}_0\mathbf{r}]$ to uniform random vectors over $\mathbb{G}^{2k}$. For the next step first note that a $\mathcal{U}_{2k,k}$-MDDH challenge $([\mathbf{A}_0], [\mathbf{v}])$ can be efficiently transformed into a $\mathcal{U}_{2k,k}$-MDDH challenge $([\mathbf{A}_1], [\mathbf{v}'])$, such that a real MDDH challenge $[\mathbf{v}] = [\mathbf{A}_0\mathbf{r}]$ is transformed into $[\mathbf{v}'] = [\mathbf{A}_1\mathbf{r}]$, and a uniform $[\mathbf{v}]$ is transformed into a uniform $[\mathbf{v}']$.

This is obtained simply by picking $\mathbf{U} \leftarrow_R \mathbb{Z}_p^{k\times k}$ and defining $[\mathbf{A}_1]$ as $\overline{[\mathbf{A}_1]} := \overline{[\mathbf{A}_0]}$, $\underline{[\mathbf{A}_1]} := \mathbf{U}\underline{[\mathbf{A}_0]}$, $\overline{[\mathbf{v}']} := \overline{[\mathbf{v}]}$, and $\underline{[\mathbf{v}']} := \mathbf{U}\underline{[\mathbf{v}]}$. With probability $1 - k \cdot 2^{-\Omega(\lambda)}$ over the choices of $\mathbf{A}_0 \leftarrow_R \mathcal{U}_{2k,k}$, $\underline{\mathbf{A}}_0$ is full rank, and $\mathbf{U}\underline{\mathbf{A}}_0$ is uniformly random over $\mathbb{Z}_p^{k\times k}$.

Given $([\mathbf{A}_0], [\mathbf{v}])$, we can compute the tag $\tau := \mathsf{H}(\overline{[\mathbf{v}]})$ and, depending on τ_{i+1}, decide whether we have to switch to $([\mathbf{A}_1], [\mathbf{v}'])$. Note that this does not affect the tag, as it only depends on $\overline{[\mathbf{v}]}$. Now applying the Q_{enc}-fold $\mathcal{U}_{2k,k}$-MDDH a second time allows to change to challenge ciphertexts of the form $[\mathbf{A}_{\tau_{i+1}}\mathbf{r}]$ as desired. Further note that simulating $\mathcal{O}_{\mathrm{dec}}$ only requires knowing $\mathbf{A}^\perp$, which is independent of $\mathbf{A}_0$ and $\mathbf{A}_1$, and therefore, does not compromise the $\mathcal{U}_{2k,k}$-MDDH assumption with respect to those matrices.

Finally, employing Theorem 6 (random self-reducibility of the Q_{enc}-fold $\mathcal{U}_{2k,k}$-MDDH assumption) and Theorem 5 ($\mathcal{D}_{2k,k}$-MDDH $\Rightarrow \mathcal{U}_{2k,k}$-MDDH), we obtain an adversary $\mathcal{B}$ such that $T(\mathcal{B}) \approx T(\mathcal{A}) + (Q_{\mathrm{enc}} + Q_{\mathrm{dec}}) \cdot \mathsf{poly}(\lambda)$ for a polynomial poly independent of $T(\mathcal{A})$, and such that

$$|\varepsilon_{\mathbf{H}.4.i.0} - \varepsilon_{\mathbf{H}.4.i.1}| \leq 2 \cdot \mathrm{Adv}^{\mathrm{mddh}}_{\mathbb{G},\mathcal{D}_{2k,k},\mathcal{B}}(\lambda) + \frac{2}{p-1}.$$

Transition $\mathbf{H}_{4.i.1} \rightsquigarrow \mathbf{H}_{4.i.2}$: For $i = 0, \ldots, \lambda-1$, the change introduced in $\mathbf{H}_{4.i.2}$ is that $\mathcal{O}_{\mathrm{dec}}(\mathsf{pred}, ([\mathbf{t}], \Pi))$ checks whether $[\mathbf{t}] \in \mathcal{L}^{\mathrm{ext}}$ (note that this can be checked efficiently given $\mathbf{A}_0$ and $\mathbf{A}_1$). If this is the case, $\mathcal{O}_{\mathrm{dec}}$ continues as

in $\mathbf{H}_{4.i.1}$, otherwise, it returns $\perp$. This change can only be detected if the adversary $\mathcal{A}$ manages to submit a valid decryption query with $[\mathbf{t}] \notin \mathcal{L}^{\text{ext}}$. We bound this event by constructing an adversary $\mathcal{B}$ from $\mathcal{A}$ attacking the constrained $\mathcal{L}^{\text{ext}}$-soundness of PS'.

On receiving the public parameters ppk of the proof system, $\mathcal{B}$ chooses $\mathbf{k}_0, \mathbf{k}_1 \leftarrow \mathbb{Z}_p^{2k}$ and sends the public key $\mathsf{pk} := (\mathsf{ppk}, [\mathbf{k}_0^\top \mathbf{A}], [\mathbf{k}_1^\top \mathbf{A}])$ to $\mathcal{A}$.

For answering encryption queries of $\mathcal{A}$, the adversary $\mathcal{B}$ first employs its simulation oracle to obtain $([\mathbf{t}], \Pi, [\kappa])$. Recall that $\mathcal{O}_{\text{sim}}$ of PS returns challenges with $[\mathbf{t}] \in \text{span}([\mathbf{A}_0]) \cup \text{span}([\mathbf{A}_1])$. The adversary then computes $\tau := \mathsf{H}(\overline{[\mathbf{t}]})$ and if $[\mathbf{t}] \notin \text{span}([\mathbf{A}_{\tau_{i+1}}])$ it rejects and queries the simulation oracle again. As $\overline{[\mathbf{A}_0]} = \overline{[\mathbf{A}_1]}$, τ_{i+1} is independent of the span in which $[\mathbf{t}]$ lies. Therefore $\mathcal{B}$ rejects with probability merely $1/2$ and thus requires only $\mathsf{poly}(\lambda) \in O(\lambda)$ time to obtain a query of the desired form with probability $2^{-\Omega(\lambda)}$ (otherwise it aborts), where poly is a polynomial independent of $T(\mathcal{A})$. Finally $\mathcal{B}$ sets $C := ([\mathbf{t}], \Pi)$ and $K := (\mathbf{k}_0 + \tau \mathbf{k}_1 + \mathbf{F}_i(\tau_{|i}))^\top [\mathbf{t}] + [\kappa]$ and returns (C, K) to $\mathcal{A}$.

To answer a decryption query $(\mathsf{pred}, ([\mathbf{t}], \Pi))$ the adversary $\mathcal{B}$ has to query its verification oracle for each distinct value $\mathbf{F}_i(\tau_{|i}^{(j)})$, where $\tau^{(j)} \in \mathcal{Q}_{\text{enc}}$, until the simulation oracle replies something other than $\perp$. Note that $\mathbf{F}_i$ can take at most 2^i values, so for small i the number of simulation queries will be much less than Q_{enc} in general. Nevertheless to keep the bound simpler, we will bound the total running time of the adversary $\mathcal{B}$ to answer decryption queries by $Q_{\text{dec}} \cdot Q_{\text{enc}} \cdot \mathsf{poly}(\lambda)$, where poly is a polynomial independent of $T(\mathcal{A})$.

Namely, on a decryption query $(\mathsf{pred}, ([\mathbf{t}], \Pi))$, the adversary $\mathcal{B}$ computes the tag $\tau^\star := \mathsf{H}(\overline{[\mathbf{t}]})$ as usual and defines for all $\tau^{(j)} \in \mathcal{Q}_{\text{enc}}$ with distinct images $\mathbf{F}_i(\tau_{|i}^{(j)})$ additional predicates $\mathsf{pred}_j \colon \mathbb{G} \to \{0,1\}, K \mapsto \mathsf{pred}\left((\mathbf{k}_0 + \tau^\star \mathbf{k}_1 + \mathbf{F}_i(\tau_{|i}^{(j)}))^\top [\mathbf{t}] + K\right)$. Then for each $j \in [|\mathcal{Q}_{\text{enc}}|]$ adversary $\mathcal{B}$ queries $([\mathbf{t}], \Pi, \mathsf{pred}_j)$ to its verification oracle $\mathcal{O}_{\text{ver}}$, and does the following.

In case $[\mathbf{t}] \in \text{span}([\mathbf{A}])$, the oracle $\mathcal{O}_{\text{ver}}$ returns either $\perp$ or a key $[\kappa]$. In the former case $\mathcal{B}$ forwards $\perp$ to $\mathcal{A}$, in the latter the key $K := (\mathbf{k}_0 + \tau^\star \mathbf{k}_1)^\top [\mathbf{t}] + [\kappa]$.

In case $[\mathbf{t}] \in \text{span}([\mathbf{A}_0]) \cup \text{span}([\mathbf{A}_1])$, $\mathcal{O}_{\text{ver}}$ either returns $\perp$, or the adversary $\mathcal{B}$ loses the constrained soundness game. In case $\mathcal{B}$ has not lost, it forwards $\perp$ to $\mathcal{A}$. Otherwise $\mathcal{A}$ managed to submit a critical query in respect to both games $\mathbf{H}_{4.i.1}$ and $\mathbf{H}_{4.i.2}$ and did thus not succeed in distinguishing between the two.

Finally, in case $[\mathbf{t}] \notin \mathcal{L}^{\text{ext}}$, $\mathcal{O}_{\text{ver}}$ either returns $\perp$, which $\mathcal{B}$ forwards to $\mathcal{A}$, or it returns "win" to $\mathcal{B}$. Note that only in this case $\mathcal{A}$ managed to submit a valid query outside $\mathcal{L}^{\text{snd}}$ and therefore managed to distinguish between the two games.

Altogether we obtain an adversary $\mathcal{B}$ breaking $\mathcal{L}^{\text{ext}}$-constrained soundness in time $T(\mathcal{B}) \approx T(\mathcal{A}) + (Q_{\text{enc}} + Q_{\text{enc}} \cdot Q_{\text{dec}}) \cdot \mathsf{poly}(\lambda)$, where poly is a

$\underline{\mathcal{O}_{\text{enc}}}$:
$\mathbf{r} \leftarrow_R \mathbb{Z}_p^k$
$[\mathbf{t}] := [\mathbf{A}_0]\mathbf{r}$
$\tau := \mathsf{H}(\overline{[\mathbf{t}]})$
$(\Pi, [\kappa]) := \mathsf{PSim}'(\mathsf{ppk}, \mathsf{psk}, [\mathbf{t}])$
$C := ([\mathbf{t}], \Pi)$
$K := \left(\mathbf{k}_0 + \tau\mathbf{k}_1 + \mathbf{k}_\Delta^{\text{enc}}(\tau)\right)^\top [\mathbf{t}] + [\kappa]$
$\mathcal{C}_{\text{enc}} := \mathcal{C}_{\text{enc}} \cup \{C\}$
return (C, K)

$\underline{\mathcal{O}_{\text{dec}}(\mathsf{pred}, ([\mathbf{t}], \Pi))}$:
$(v, [\kappa]) := \mathsf{PVer}'(\mathsf{psk}, [\mathbf{t}], \Pi)$
$\tau^\star := \mathsf{H}(\overline{[\mathbf{t}]})$
if $([\mathbf{t}], \Pi) \notin \mathcal{C}_{\text{enc}}$ **and** $v = 1$ **and** τ is fresh
  if $[\mathbf{t}] \in \text{span}([\mathbf{A}])$
    $K := (\mathbf{k}_0 + \tau^\star\mathbf{k}_1)^\top [\mathbf{t}] + [\kappa]$
    if $\mathsf{pred}(K) = 1$
      return K
  else if $[\mathbf{t}] \in \text{span}([\mathbf{A}_0]) \cup \text{span}([\mathbf{A}_1])$
    $\mathcal{S}_K := \left\{ \left(\mathbf{k}_0 + \tau^\star\mathbf{k}_1 + \mathbf{k}_\Delta^{\text{dec}}(\tau, [\mathbf{t}])\right)^\top [\mathbf{t}] + [\kappa] \;\middle|\; \tau \in \mathcal{Q}_{\text{enc}} \right\}$
    if $\exists K \in \mathcal{S}_K$ **such that** $\mathsf{pred}(K) = 1$
      return *critical query* **and abort**
return $\perp$

Figure 3.12: Oracles in Game $\mathbf{H}_{4.i.3}$

polynomial independent of $T(\mathcal{A})$, such that

$$|\varepsilon_{\mathbf{H}.4.i.1} - \varepsilon_{\mathbf{H}.4.i.2}| \leq \text{Adv}^{\text{snd}}_{\mathcal{L}^{\text{ext}}, \mathsf{PS}', \mathcal{B}}(\lambda) + Q_{\text{enc}} \cdot 2^{-\Omega(\lambda)}.$$

Transition $\mathbf{H}_{4.i.2} \rightsquigarrow \mathbf{H}_{4.i.3}$: As described in Fig. 3.12, game $\mathbf{H}_{4.i.3}$, the oracle $\mathcal{O}_{\text{enc}}$ computes the key using an additional summand $\mathbf{k}_\Delta^{\text{enc}}(\tau)$ for $\tau := \mathsf{H}(\overline{[\mathbf{t}]})$. Similarly, $\mathcal{O}_{\text{dec}}$ uses a vector $\mathbf{k}_\Delta^{\text{dec}}(\tau, [\mathbf{t}])$ for $\tau \in \mathcal{Q}_{\text{enc}}$. In encryption queries $\mathbf{k}_\Delta^{\text{enc}}(\tau)$ for $\tau := \mathsf{H}(\overline{[\mathbf{t}]})$ is defined as

$$\mathbf{k}_\Delta^{\text{enc}}(\tau) := \begin{cases} \mathbf{A}_0^\perp \widetilde{\mathbf{F}}_i^{(0)}(\tau_{|i}) + \mathbf{A}_1^\perp \mathbf{F}_i^{(1)}(\tau_{|i}), & \text{if } \tau_{i+1} = 0 \\ \mathbf{A}_0^\perp \mathbf{F}_i^{(0)}(\tau_{|i}) + \mathbf{A}_1^\perp \widetilde{\mathbf{F}}_i^{(1)}(\tau_{|i}), & \text{if } \tau_{i+1} = 1, \end{cases}$$

where $\mathbf{A}_0^\perp \in \text{orth}([\mathbf{A}_0])$, $\mathbf{A}_1^\perp \in \text{orth}([\mathbf{A}_1])$ and $\mathbf{F}_i^{(0)}, \mathbf{F}_i^{(1)}, \widetilde{\mathbf{F}}_i^{(0)}, \widetilde{\mathbf{F}}_i^{(1)}$: $\{0,1\}^i \to \mathbb{Z}_p^k$ are independent random functions, such that $\mathbf{F}_i(\tau_{|i}) = \mathbf{A}_0^\perp \mathbf{F}_i^{(0)}(\tau_{|i}) + \mathbf{A}_1^\perp \mathbf{F}_i^{(1)}(\tau_{|i})$. Note that with probability $1 - 2^{-\Omega(\lambda)}$ over the choices of $\mathbf{A}_0, \mathbf{A}_1$ the column vectors of $\mathbf{A}_0^\perp$ and $\mathbf{A}_1^\perp$ form a basis of $\mathbb{Z}_p^{2k}$ and thus such $\mathbf{F}_i^{(0)}$, $\mathbf{F}_i^{(1)}$ exist. Further for any bit $b \in \{0,1\}$, and $\mathbf{t} \in \text{span}(\mathbf{A}_b)$ we have

$$\mathbf{k}_\Delta^{\text{enc}}(\tau)^\top \mathbf{t} = \left(\mathbf{k}_\Delta^{\text{enc}}(\tau) + \mathbf{A}_b^\perp \widetilde{\mathbf{F}}_i^{(b)}\right)^\top \mathbf{t}.$$

Thus the change of the encryption oracle is merely conceptional.

The same holds true for the decryption oracle, where we compute the set of admissible keys depending on $[\mathbf{t}]$. Namely, for each tag $\tau \in \mathcal{Q}_{\text{enc}}$, we define $\mathbf{k}_\Delta^{\text{dec}}(\tau, [\mathbf{t}])$ as

$$\mathbf{k}_\Delta^{\text{dec}}(\tau, [\mathbf{t}]) \coloneqq \begin{cases} \mathbf{A}_0^\perp \widetilde{\mathbf{F}}_i^{(0)}(\tau_{|i}) + \mathbf{A}_1^\perp \mathbf{F}_i^{(1)}(\tau_{|i}), & \text{if } [\mathbf{t}] \in \text{span}([\mathbf{A}_0]) \\ \mathbf{A}_0^\perp \mathbf{F}_i^{(0)}(\tau_{|i}) + \mathbf{A}_1^\perp \widetilde{\mathbf{F}}_i^{(1)}(\tau_{|i}), & \text{if } [\mathbf{t}] \in \text{span}([\mathbf{A}_1]) \end{cases}$$

Therefore, $\mathbf{H}_{4.i.2}$ and $\mathbf{H}_{4.i.3}$ are identically distributed and we obtain

$$\varepsilon_{\mathbf{H}.4.i.2} = \varepsilon_{\mathbf{H}.4.i.3}.$$

Transition $\mathbf{H}_{4.i.3} \rightsquigarrow \mathbf{H}_{4.i.4}$: In game $\mathbf{H}_{4.i.4}$, for $i = 0, \ldots, \lambda - 1$ we define

$$\mathbf{F}_{i+1} \colon \{0,1\}^{i+1} \to \mathbb{Z}_p^{2k}$$

as

$$\mathbf{F}_{i+1}(\tau_{|i+1}) := \begin{cases} \mathbf{A}_0^\perp \widetilde{\mathbf{F}}_i^{(0)}(\tau_{|i}) + \mathbf{A}_1^\perp \mathbf{F}_i^{(1)}(\tau_{|i}), & \text{if } \tau_{i+1} = 0 \\ \mathbf{A}_0^\perp \mathbf{F}_i^{(0)}(\tau_{|i}) + \mathbf{A}_1^\perp \widetilde{\mathbf{F}}_i^{(1)}(\tau_{|i}), & \text{if } \tau_{i+1} = 1. \end{cases}$$

Note that this defines a random function, when $\mathbf{F}_i^{(0)}, \mathbf{F}_i^{(1)}, \widetilde{\mathbf{F}}_i^{(0)}, \widetilde{\mathbf{F}}_i^{(1)} : \{0,1\}^i \to \mathbb{Z}_p^k$ are independent random functions.

Similarly, in decryption queries for $\tau \in \mathcal{Q}_{\text{dec}}$ we use $\mathbf{F}_{i+1}$ as defined above applied to $\tau_{|i} d_{[\mathbf{t}]}$, where $d_{[\mathbf{t}]}$ is defined as

$$d_{[\mathbf{t}]} := \begin{cases} 0, & \text{if } [\mathbf{t}] \in \text{span}([\mathbf{A}_0]) \\ 1, & \text{if } [\mathbf{t}] \in \text{span}([\mathbf{A}_1]). \end{cases}$$

As the changes again are merely conceptional, we have

$$\varepsilon_{\mathbf{H}.4.i.3} = \varepsilon_{\mathbf{H}.4.i.4}.$$

Transition $\mathbf{H}_{4.i.4} \rightsquigarrow \mathbf{H}_{4.i.5}$: From game $\mathbf{H}_{4.i.5}$ on, we again allow decryption queries outside $\mathcal{L}^{\text{ext}}$. This can only increase the winning chances of the adversary, because it does not change the view on non-critical queries. We thus have

$$\varepsilon_{\mathbf{H}.4.i.4} \leq \varepsilon_{\mathbf{H}.4.i.5}.$$

Transition $\mathbf{H}_{4.i.5} \rightsquigarrow \mathbf{H}_{4.i.6}$: Game $\mathbf{H}_{4.i.6}$, for $i = 0, \ldots, \lambda - 1$, is identical to $\mathbf{H}_{4.i.5}$, except for $\mathcal{O}_{\text{dec}}$, which now computes the set of valid keys as

$$\mathcal{S}_K := \left\{ \left(\mathbf{k}_0 + \tau^\star \mathbf{k}_1 + \mathbf{F}_{i+1}(\tau_{|i} \, b) \right)^\top [\mathbf{t}] \;\middle|\; \tau \in \mathcal{Q}_{\text{enc}}, b \in \{0,1\} \right\}$$

Note that this set includes the set of keys computed in $\mathbf{H}_{4.i.5}$. Therefore, this increases the probability of the adversary to submit a critical query, while not changing its view on non-critical queries. In conclusion,

$$\varepsilon_{\mathbf{H}.4.i.5} \leq \varepsilon_{\mathbf{H}.4.i.6}.$$

Transition $\mathbf{H}_{4.i.6} \rightsquigarrow \mathbf{H}_{4.i.7}$: Game $\mathbf{H}_{4.i.7}$, for $i = 0, \ldots, \lambda - 1$, is identical to $\mathbf{H}_{4.i.6}$, except for $\mathcal{O}_{\mathrm{dec}}$, which now computes the set of valid keys as

$$\mathcal{S}_K := \left\{ \left(\mathbf{k}_0 + \tau^\star \mathbf{k}_1 + \mathbf{F}_{i+1}(\tau_{|i} \, \tau_{i+1}) \right)^\top [\mathbf{t}] \;\middle|\; \tau \in \mathcal{Q}_{\mathrm{enc}}, \right\}.$$

It suffices to show that with all but negligible probability, there is no key in $\mathcal{S}_K$ which corresponds to a tag $\tau \in \mathcal{Q}_{\mathrm{enc}}$ and a bit $b \in \{0,1\}$ such that $\tau_{|i} b \in \{0,1\}^{i+1}$ is not the prefix of any tag in $\mathcal{Q}_{\mathrm{enc}}$, and that satisfies pred. We proceed via a hybrid argument over all queries to $\mathcal{O}_{\mathrm{dec}}$. To that end, we introduce intermediate games $\mathbf{H}_{4.i.6.j}$ for $j = 0, \ldots, Q_{\mathrm{dec}}$, defined as $\mathbf{H}_{4.i.6}$, except that $\mathcal{O}_{\mathrm{dec}}$ proceeds as in game $\mathbf{H}_{4.i.7}$ on its j-th last queries. We show that:

$$\mathbf{H}_{4.i.6} = \mathbf{H}_{4.i.6.0} \approx_s \mathbf{H}_{4.i.6.1} \approx_s \ldots \approx_s \mathbf{H}_{4.i.6.Q_{\mathrm{dec}}} = \mathbf{H}_{4.i.7},$$

where by $\approx_s$ we denote statistical closeness. We show that for all $j = 0, \ldots, Q_{\mathrm{dec}} - 1$,

$$|\varepsilon_{\mathbf{H}.4.i.6.j} - \varepsilon_{\mathbf{H}.4.i.6.j+1}| \leq Q_{\mathrm{enc}} \cdot \Pr_{K \leftarrow_R \mathcal{K}}[\mathsf{pred}_{j+1}(K) = 1].$$

This is because for all tags $\tau \in \mathcal{Q}_{\mathrm{enc}}$ and $b \in \{0,1\}$ such that $\tau_{|i} b \in \{0,1\}^{i+1}$ is not prefix of any $\tau \in \mathcal{Q}_{\mathrm{enc}}$, the value $\mathbf{F}_{i+1}(\tau_{|i} b)$ is a random value, uniform over $\mathbb{Z}_p^k$, independent of $\mathcal{A}$'s view before its $(j+1)$-st query to $\mathcal{O}_{\mathrm{dec}}$. Summing up, we obtain

$$|\varepsilon_{\mathbf{H}.4.i.6} - \varepsilon_{\mathbf{H}.4.i.7}| \leq Q_{\mathrm{enc}} \cdot Q_{\mathrm{dec}} \cdot \mathrm{uncert}_{\mathcal{A}}(\lambda).$$

Transition $\mathbf{H}_{4.i.7} \rightsquigarrow \mathbf{H}_{4.(i+1).0}$: For $i = 0, \ldots, \lambda - 1$, in $\mathbf{H}_{4.(i+1).0}$ the challenge ciphertexts are switched back to the span of $[\mathbf{A}_0]$ independent of the tag τ, the transition is thus the reverse to $\mathbf{H}_{4.i.0} \rightsquigarrow \mathbf{H}_{4.i.1}$. More precisely, we first tightly switch all challenges of the form $[\mathbf{A}_{\tau_{i+1}} \mathbf{r}]$ to uniform random vectors over $\mathbb{G}^{2k}$ and then back to vectors in the span of $[\mathbf{A}_0]$. From an adversary $\mathcal{A}$ detecting this change, we can construct an adversary $\mathcal{B}$ attacking the Q_{enc}-fold $\mathcal{U}_{2k,k}$-MDDH assumption as follows. On input $([\mathbf{A}_0], [\mathbf{v}_1 | \cdots | \mathbf{v}_{Q_{\mathrm{enc}}}])$ with $[\mathbf{A}_0] \in \mathbb{G}^{2k \times k}$ and $[\mathbf{V}] := [\mathbf{v}_1 | \cdots | \mathbf{v}_{Q_{\mathrm{enc}}}] \in \mathbb{G}^{2k \times Q_{\mathrm{enc}}}$, the adversary $\mathcal{B}$ chooses $\mathbf{U} \leftarrow \mathbb{Z}_p^{k \times k}$ and sets $[\mathbf{A}_1]$ such that $\overline{[\mathbf{A}_1]} = \overline{[\mathbf{A}_0]}$ and $\underline{[\mathbf{A}_1]} = \mathbf{U}\underline{[\mathbf{A}_0]}$. With probability $1 - k \cdot 2^{-\Omega(\lambda)}$ over the choices of $\mathbf{A}_0 \leftarrow_R \mathcal{U}_{2k,k}$, $\underline{\mathbf{A}_0}$ is full rank, and $\mathbf{U}\underline{\mathbf{A}_0}$ is uniformly random over $\mathbb{Z}_p^{k \times k}$.

Further $\mathcal{B}$ chooses the rest of the public parameters as in Section 3.2 and generates the public and secret keys of the KEM by invoking KGen on input 1^λ. On the j-th query of $\mathcal{A}$ to $\mathcal{O}_{\mathrm{enc}}$, $\mathcal{B}$ computes $\tau := \mathsf{H}(\overline{[\mathbf{v}_j]})$. In case $\tau_{i+1} = 0$, the adversary continues answering the decryption query with $\underline{[\mathbf{t}]} := \underline{[\mathbf{v}_j]}$. In case $\tau_{i+1} = 1$, the adversary instead sets $[\mathbf{t}]$ such that $\overline{[\mathbf{t}]} = \overline{[\mathbf{v}_j]}$ and $\underline{[\mathbf{t}]} = \mathbf{U}\underline{[\mathbf{v}_j]}$. In case $[\mathbf{V}]$ was uniformly random over $\mathbb{G}^{2k \times Q_{\mathrm{enc}}}$, the adversary $\mathcal{B}$ simulates the intermediary game, where all challenge ciphertexts are chosen uniformly random. If instead for each

$j \in \{1, \ldots, Q_{\text{enc}}\}$ there exists an $\mathbf{r}_j \in \mathbb{Z}_p^k$ such that $[\mathbf{v}_j] = [\mathbf{A}_0]\mathbf{r}_j$, the adversary simulates game $\mathbf{H}_{4.i.7}$, as in this case for all $j \in \{1, \ldots, Q_{\text{enc}}\}$ we have $[\mathbf{t}_j] = [\mathbf{A}_{\tau_{i+1}}\mathbf{r}_j]$.

Now we can employ the Q_{enc}-fold $\mathcal{U}_{2k,k}$-MDDH assumption a second time to tightly switch back the challenge ciphertexts from random to the span of $[\mathbf{A}_0]$.

Finally, using Theorem 6 (random self-reducibility of the $\mathcal{U}_{2k,k}$-MDDH assumption) and Theorem 5 ($\mathcal{D}_{2k,k}$-MDDH $\Rightarrow$ $\mathcal{U}_{2k,k}$-MDDH), we obtain an adversary $\mathcal{B}'$ such that $T(\mathcal{B}') \approx T(\mathcal{A}) + (Q_{\text{enc}} + Q_{\text{dec}}) \cdot \mathsf{poly}(\lambda)$ for a polynomial poly independent of $T(\mathcal{A})$, and such that

$$|\varepsilon_{\mathbf{H}.4.i.5} - \varepsilon_{\mathbf{H}.4.(i+1).0}| \leq 2 \cdot \mathrm{Adv}^{\mathrm{mddh}}_{\mathbb{G},\mathcal{D}_{2k,k},\mathcal{B}'}(\lambda) + \frac{2}{p-1}.$$

Game $\mathbf{H}_5$: We now show that an adversary has only negligible chances to win $\mathbf{H}_5 \coloneqq \mathbf{H}_{4.\lambda.7}$. We argue as follows.

First, for $\mathbf{u} \leftarrow_R \mathbb{Z}_p^k$ the tuples

$$\left(\mathbf{k}_1, \left(\mathbf{F}_\lambda(\tau)\right)_{\tau\in\{0,1\}^\lambda}\right) \text{ and } \left(\mathbf{k}_1 - \mathbf{A}^\perp\mathbf{u}, \left(\mathbf{F}_\lambda(\tau) + \tau\mathbf{A}^\perp\mathbf{u}\right)_{\tau\in\{0,1\}^\lambda}\right)$$

are distributed identically.

Second, the set of tags computed by $\mathcal{O}_{\text{enc}}$ and the set of tags computed by $\mathcal{O}_{\text{dec}}$ are disjoint (recall that we established this in game $\mathbf{G}_1$ in the proof of Theorem 38).

Note that $\mathbf{u}$ does not show up when $\mathcal{O}_{\text{enc}}$ computes challenge keys, since in this case

$$\begin{aligned} K &= \left(\mathbf{k}_0 + \tau\left(\mathbf{k}_1 - \mathbf{A}^\perp\mathbf{u}\right) + \mathbf{F}_\lambda(\tau) + \tau\mathbf{A}^\perp\mathbf{u}\right)^\top [\mathbf{t}] \\ &= \left(\mathbf{k}_0 + \tau\mathbf{k}_1 + \mathbf{F}_\lambda(\tau)\right)^\top [\mathbf{t}], \end{aligned}$$

that is, the extra terms cancel each other out.

On the contrary, an extra term appears when $\mathcal{O}_{\text{dec}}$ is queried on an input that contains $[\mathbf{t}]$ such that $\mathbf{t}^\top\mathbf{A}^\perp \neq 0$, since $\mathcal{O}_{\text{dec}}$ computes $\tau^\star \coloneqq \mathsf{H}(\overline{[\mathbf{t}]})$ and the set of keys as

$$\mathcal{S}_K \coloneqq \left\{ \left(\mathbf{k}_0 + \tau^\star\mathbf{k}_1 + \mathbf{F}_\lambda(\tau) + (\tau^\star - \tau)\mathbf{A}^\perp\mathbf{u}\right)^\top [\mathbf{t}] \;\middle|\; \tau \in \mathcal{Q}_{\text{enc}} \right\}.$$

As we require tags to be fresh, we have $\tau^\star \notin \mathcal{Q}_{\text{enc}}$ and therefore the term $(\tau^\star - \tau)(\mathbf{A}^\perp\mathbf{u})^\top\mathbf{t}$ is uniformly random over $\mathbb{Z}_p$. Thus, the marginal distribution of each key in $\mathcal{S}_K$ is uniform over $\mathbb{G}$. Using a hybrid argument over all queries to $\mathcal{O}_{\text{dec}}$, we hence obtain

$$|\varepsilon_{\mathbf{H}.5}| \leq Q_{\text{dec}} \cdot Q_{\text{enc}} \cdot \mathrm{uncert}_{\mathcal{A}}(\lambda).$$

□

Chapter 4

Structure-Preserving Signatures

Building on the techniques presented in the previous chapter, we continue with giving a tightly secure structure-preserving signature scheme. Recall that a signature scheme is *structure-preserving* if messages and signatures consist of group elements only, and its signing and verification algorithm can be expressed as (pairing) equations over a cyclic group.

Overview of our Strategy. The main idea towards our construction is to distill the proof technique of [Hof17, GHK17] in a core lemma, which makes it more easily transferable to the context of signatures. Building on the core lemma, we construct a tightly secure (but not yet structure-preserving) message authentication code. This is similar to the approach to constructing a tightly secure identity based encryption scheme by Blazy, Kiltz, and Pan [BKP14]. To convert the message authentication code we adapt the generic transformation of Bellare and Goldwasser [BG90]. This yields an *almost* structure-preserving signature scheme, but where the message is an element in $\mathbb{Z}_p$ instead of a group element. We solve this by using a slightly different equation for computing signatures. A similar strategy was used by Kiltz, Pan, and Wee [KPW15] to construct structure-preserving signatures, but not in the context of tight security.

Our Core Lemma. In the core lemma we follow the randomization strategy called "adaptive partitioning" introduced by [Hof17], but with significantly smaller security loss of only $\mathcal{O}(\log Q)$ instead of $\mathcal{O}(\lambda)$, where Q is the number of queries made by the adversary. The idea of the core lemma is as follows: Let $\mathcal{L}_0 := \{[\mathbf{a}_0] \cdot r \mid r \in \mathbb{Z}_p\}$ and $\mathcal{L}_1 := \{[\mathbf{a}_1] \cdot r \mid r \in \mathbb{Z}_p\}$ be two (public) languages where $[\mathbf{a}_0], [\mathbf{a}_1] \in \mathbb{G}^2$. Let $\mathbf{k}_0 \in \mathbb{Z}_p^2$ be a hash proof system secret key and let statements be of the form

$$([\mathbf{t}], [\mathbf{k}_0^\top \mathbf{t}], \Pi),$$

where $[\mathbf{t}] \in \mathcal{L}_0$ and Π is a proof of well-formedness (that is $[\mathbf{t}] \in \mathcal{L}_0 \cup \mathcal{L}_1$).

Then, the core lemma allows to gradually randomize the secret key $\mathbf{k}_0$, such that finally each statement has a freshly randomized secret key. In the proof of security for the message authentication code and signature schemes this will allow to finally argue (using a purely information-theoretic argument) that an adversary has only negligible chance to provide a valid verification query on a fresh message.

The adversary we consider in the core lemma can query evaluations for random statements and ask for validity of statements. The adversary wins, if the verification oracle returns *valid* on a fresh statement.

Using the techniques of [Hof17, GHK17] we show that the randomization of the secret key can only increase the chances of the adversary to win. The idea is in each step to switch half of the statements to $\mathcal{L}_1$. Then, a fresh random offset is added to all statements in $\mathcal{L}_0$, and another fresh random offset to all statements

in $\mathcal{L}_1$, such that the change is not detectable on evaluation queries. Similar to [Hof17, GHK17] we cannot ensure that the adversary follows the partitioning, but by the proof of well-formedness we can reject any verification query with statement outside $\mathcal{L}_0 \cup \mathcal{L}_1$, which will be sufficient for the proof to go through. Our security loss is only $\mathcal{O}(\log Q)$, because—different to [Hof17, GHK17]—we enumerate the queries and partition according to the i-th bit of the query number (instead of the hash of the statement).

Note that here the disjunction of two languages $\mathcal{L}_0 \cup \mathcal{L}_1$ is sufficient, because the adversary does not have access to the hash proof system public key itself. Even in our signature scheme, a sort of "public commitment" to the hash proof system secret key is sufficient to verify signatures. In the context of encryption schemes, on the other hand, the hash proof system public key is necessary for encryption and thus has to be part of the public key. Therefore, randomization of the secret key can take place only "outside" the basic language $\mathcal{L}$.

Our Tightly Secure MAC. Recall that for our encryption scheme of the previous scheme, we derive the key for symmetric authenticated encryption on a statement $[\mathbf{t}] \in \mathcal{L}_0$ as

$$[k] = \mathbf{k}_0^\top [\mathbf{t}] + \mu \cdot \mathbf{k}_1^\top [\mathbf{t}],$$

where $\mathbf{k}_0, \mathbf{k}_1 \in \mathbb{Z}_p^2$ and $\mu = H([\mathbf{t}])$ for a collision resistant hash function $H \colon \mathbb{G}^2 \to \mathbb{Z}_p$.

By a generic construction of Dodis et al. [DKPW12] replacing μ by the message $M \in \mathbb{Z}_p$ to be authenticated yields a message authentication code. Instead of directly reducing the security of the MAC to our key encapsulation mechanism, we use the core lemma to prove security. As mentioned this allows us to use a proof for the language $[\mathbf{t}] \in \mathcal{L}_0 \cup \mathcal{L}_1$ (without requiring extensibility to a third language).

Our Tightly Secure Signatures. To make the MAC publicly verifiable we use an optimized version of the generic transformation of Bellare and Goldwasser [BG90]. Roughly, a public commitment to the secret key is added to the public key, which allows to publicly verify validity via a pairing product equation.

Further, the proof for well-formedness has to be publicly verifiable. For this purpose we use a proof by Ràfols [Ràf15], which is based on the idea of Groth, Ostrovsky and Sahai [GOS12]: To prove the disjunction of languages $\mathcal{L}_0, \mathcal{L}_1$, one gives a proof for $[\mathbf{t}] \in \mathcal{L}_0$ *and* a proof for $[\mathbf{t}] \in \mathcal{L}_1$, where one of the proofs is simulated (that is, one proof can be computed without knowledge of the witness). The public setup ensures that at least on of the statements has to be proven *honestly*.

For structure-preserving signatures we have to replace the equation

$$[k] = \mathbf{k}_0^\top [\mathbf{t}] + \mu \cdot \mathbf{k}_1^\top [\mathbf{t}]$$

such that we can sign messages $[M] \in \mathbb{G}$ *and* still apply the core lemma for the proof of security. Our new equation will be of the form

$$[k] = \mathbf{k}_0^\top [\mathbf{t}] + \mathbf{k}_1^\top \begin{bmatrix} M \\ 1 \end{bmatrix}.$$

As the first part is not affected, we can still apply the core lemma to randomize $\mathbf{k}_0$ and finally prove security using the pairwise independence of the second part of the equation (as a function of $[M]$), as for the MAC and signature scheme.

Roadmap. We start by providing the formal definition of non-interactive zero-knowledge proofs and giving an instantiation based on [GOS12, Ràf15] in Section 4.1. In Section 4.2 we present our core lemma and in Section 4.3 we show how it yields a tightly-secure message authentication code. In Section 4.4 we turn the message authentication code into a signature by making the evaluation of the hash proof system publicly verifiable. Finally, in Section 4.5 we provide our structure-preserving signature scheme and explain how our construction can be transformed into a signature scheme in the bilateral setting.

The following is taken verbatim (with minor changes) from our work [GHKP18].

4.1 A Publicly Verifiable Proof for Or-Languages

As signatures have to be *publicly verifiable*, this has to hold true also for the proof of consistency. We therefore build on *non-interactive zero-knowledge proofs*, as introduced in [BFM88].

Non-interactive zero-knowledge proofs. In the following we present the definition from [GS08].

Definition 40 (Non-interactive zero-knowledge proof [GS08])**.** We consider a family of languages $\mathcal{L} = \{\mathcal{L}_{\mathsf{pars}}\}$ with efficiently computable witness relation $\mathcal{R}_{\mathcal{L}}$. A *non-interactive zero-knowledge proof (NIZK)* for $\mathcal{L}$ is a tuple of PPT algorithms $\mathsf{PS} := (\mathsf{PGen}, \mathsf{PTGen}, \mathsf{PPrv}, \mathsf{PVer}, \mathsf{PSim})$ such that:

$\mathsf{PGen}(1^\lambda, \mathsf{pars})$: On input of the security parameter λ and the public parameters, generates a common reference string crs.

$\mathsf{PTGen}(1^\lambda, \mathsf{pars})$: On input of the security parameter λ and the public parameters, generates a common reference string crs and additionally a trapdoor td.

$\mathsf{PPrv}(\mathsf{crs}, x, w)$: Given the common reference string crs, a word $x \in \mathcal{L}$ and a witness w with $\mathcal{R}_{\mathcal{L}}(x, w) = 1$, outputs a proof $\Pi \in \mathcal{P}$.

$\mathsf{PVer}(\mathsf{crs}, x, \Pi)$: On input crs, $x \in \mathcal{L}$ and Π outputs a verdict $b \in \{0, 1\}$.

$\mathsf{PSim}(\mathsf{crs}, \mathsf{td}, x)$: Given a crs with corresponding trapdoor td and a word $x \in \mathcal{L}$, outputs a proof Π.

Further we require the following properties to hold.

Completeness: For all possible public parameters pars, all $\lambda \in \mathbb{N}$, all words $x \in \mathcal{L}$, and all witnesses w such that $\mathcal{R}_{\mathcal{L}}(x, w) = 1$, we have

$$\Pr[\mathsf{PVer}(\mathsf{crs}, x, \Pi) = 1] = 1,$$

where the probability is taken over $\mathsf{crs} \leftarrow \mathsf{PGen}\,(1^\lambda, \mathsf{pars})$ and $\Pi \leftarrow \mathsf{PPrv}(\mathsf{crs}, x, w)$.

Composable zero-knowledge: For all PPT adversaries $\mathcal{A}$ we have that

$$\begin{aligned}\mathrm{Adv}^{\mathrm{keygen}}_{\mathsf{PS},\mathcal{A}}(\lambda) := \Big|&\Pr[\mathcal{A}(1^\lambda, \mathsf{crs}) = 1 \mid \mathsf{crs} \leftarrow \mathsf{PGen}(1^\lambda, \mathsf{pars})] \\ &- \Pr[\mathcal{A}(1^\lambda, \mathsf{crs}) = 1 \mid (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{PTGen}(1^\lambda, \mathsf{pars})]\Big| \leq \mathsf{negl}(\lambda).\end{aligned}$$

$\underline{\mathsf{PGen}(1^\lambda, \mathsf{pars})}$:
$\mathbf{D} \leftarrow_R \mathcal{D}_k$, $\mathbf{z} \leftarrow_R \mathbb{Z}_p^{k+1} \backslash \mathrm{span}(\mathbf{D})$
//recall $\mathcal{D}_k := \mathcal{D}_{k+1,k}$
$\mathsf{crs} := (\mathsf{pars}, [\mathbf{D}]_2, [\mathbf{z}]_2)$
return crs

$\underline{\mathsf{PPrv}(\mathsf{crs}, [\mathbf{x}]_1, \mathbf{r})}$:
let $j \in \{0,1\}$ **s.t.** $[\mathbf{x}]_1 = [\mathbf{A}_j]_1 \cdot \mathbf{r}$
$\mathbf{v} \leftarrow_R \mathbb{Z}_p^k$
$[\mathbf{z}_{1-j}]_2 := [\mathbf{D}]_2 \cdot \mathbf{v}$
// $([\mathbf{D}]_2, [\mathbf{z}_{1-j}]_2)$ trapdoor crs
$[\mathbf{z}_j]_2 := [\mathbf{z}]_2 - [\mathbf{z}_{1-j}]_2$
// crs guaranteeing soundness
$\mathbf{S}_0, \mathbf{S}_1 \leftarrow_R \mathbb{Z}_p^{k \times k}$
$[\mathbf{C}_j]_2 := \mathbf{S}_j \cdot [\mathbf{D}]_2^\top + \mathbf{r} \cdot [\mathbf{z}_j]_2^\top$
//commitment to $\mathbf{r}$ with rand. $\mathbf{S}_j$
$[\Pi_j]_1 := [\mathbf{A}_j]_1 \cdot \mathbf{S}_j$
//proof for $\mathbf{x} = \mathbf{A}_j \mathbf{r}$
$[\mathbf{C}_{1-j}]_2 := \mathbf{S}_{1-j} \cdot [\mathbf{D}]_2^\top$
//commitment to $\mathbf{0}$ with rand. $\mathbf{S}_{1-j}$
$[\Pi_{1-j}]_1 := [\mathbf{A}_{1-j}]_1 \cdot \mathbf{S}_{1-j} - [\mathbf{x}]_1 \cdot \mathbf{v}^\top$
//trapdoor proof for $\mathbf{x} = \mathbf{A}_{1-j}\mathbf{r}$
return $([\mathbf{z}_0]_2, ([\mathbf{C}_i]_2, [\Pi_i]_1)_{i \in \{0,1\}})$

$\underline{\mathsf{PTGen}(1^\lambda, \mathsf{pars})}$:
$\mathbf{D} \leftarrow_R \mathcal{D}_k$, $\mathbf{u} \leftarrow_R \mathbb{Z}_p^k$
$\mathbf{z} := \mathbf{D} \cdot \mathbf{u}$
$\mathsf{crs} := (\mathsf{pars}, [\mathbf{D}]_2, [\mathbf{z}]_2)$, $\mathsf{td} := \mathbf{u}$
return $(\mathsf{crs}, \mathsf{td})$

$\underline{\mathsf{PVer}(\mathsf{crs}, [\mathbf{x}]_1, ([\mathbf{z}_0]_2, ([\mathbf{C}_i]_2, [\Pi_i]_1)_{i \in \{0,1\}}))}$:
$[\mathbf{z}_1]_2 := [\mathbf{z}]_2 - [\mathbf{z}_0]_2$
if for all $i \in \{0,1\}$ **it holds**
$e([\mathbf{A}_i]_1, [\mathbf{C}_i]_2)$
$= e([\Pi_i]_1, [\mathbf{D}]_2^\top) + e([\mathbf{x}]_1, [\mathbf{z}_i]_2^\top)$
//check $\mathbf{A}_i \cdot \mathbf{C}_i \stackrel{?}{=} \Pi_i \cdot \mathbf{D}^\top + \mathbf{x} \cdot \mathbf{z}_i^\top$
return 1
else return 0

$\underline{\mathsf{PSim}(\mathsf{crs}, \mathsf{td}, [\mathbf{x}]_1)}$:
parse $\mathsf{td} =: \mathbf{u}$
$\mathbf{v} \leftarrow_R \mathbb{Z}_p^k$
$[\mathbf{z}_0]_2 := [\mathbf{D}]_2 \cdot \mathbf{v}$
$[\mathbf{z}_1]_2 := [\mathbf{z}]_2 - [\mathbf{z}_0]_2$
$\mathbf{S}_0, \mathbf{S}_1 \leftarrow_R \mathbb{Z}_p^{k \times k}$
$[\mathbf{C}_0]_2 := \mathbf{S}_0 \cdot [\mathbf{D}]_2^\top$
$[\Pi_0]_1 := [\mathbf{A}_0]_1 \cdot \mathbf{S}_0 - [\mathbf{x}]_1 \cdot \mathbf{v}^\top$
$[\mathbf{C}_1]_2 := \mathbf{S}_1 \cdot [\mathbf{D}]_2^\top$
$[\Pi_1]_1 := [\mathbf{A}_1]_1 \cdot \mathbf{S}_1 - [\mathbf{x}]_1 \cdot (\mathbf{u} - \mathbf{v})^\top$
return $([\mathbf{z}_0]_2, ([\mathbf{C}_i]_2, [\Pi_i]_1)_{i \in \{0,1\}})$

Figure 4.1: NIZK argument for $\mathcal{L}^{\mathrm{snd}}$ ([GOS12, Ràf15]).

Further, for all $x \in \mathcal{L}$ with witness w such that $\mathcal{R}_{\mathcal{L}}(x, w) = 1$, the following are identically distributed:

$$\mathsf{PPrv}(\mathsf{crs}, x, w) \text{ and } \mathsf{PSim}(\mathsf{crs}, \mathsf{td}, x),$$

where $(\mathsf{crs}, \mathsf{td}) \leftarrow_R \mathsf{PTGen}(1^\lambda)$.

Perfect soundness: For all crs in the range of $\mathsf{PGen}(1^\lambda, \mathsf{pars})$, for all words $x \notin \mathcal{L}$ and all proofs Π it holds $\mathsf{PVer}(\mathsf{crs}, x, \Pi) = 0$.

In the following, we recall an instantiation of a NIZK for an or-language first given in [GOS12] and later generalized in [Ràf15] for more general languages. This NIZK will be a crucial part of all our constructions, allowing to employ the randomization techniques from [AHN+17, GHK17, Hof17] to obtain a tight security reduction.

Public Parameters. Let $\mathcal{PG} \leftarrow \mathsf{BGen}(1^\lambda)$. Let $k \in \mathbb{N}$. Let $\mathbf{A}_0, \mathbf{A}_1 \leftarrow_R \mathcal{D}_{2k,k}$. We define the public parameters to comprise

$$\mathsf{pars} := (\mathcal{PG}, [\mathbf{A}_0]_1, [\mathbf{A}_1]_1).$$

We consider $k \in \mathbb{N}$ to be chosen ahead of time, fixed and implicitly known to all algorithms (recall that in practice, $k = 1$ for SXDH, $k = 2$ for DLIN).

Or-Proof ([GOS12, Ràf15]). In Figure 4.1 we present a non-interactive zero-knowledge proof for the OR-language

$$\mathcal{L}^{\mathrm{snd}} := \{[\mathbf{x}]_1 \in \mathbb{Z}_p^{2k} \mid \exists \mathbf{r} \in \mathbb{Z}_p^k \colon [\mathbf{x}]_1 = [\mathbf{A}_0]_1 \cdot \mathbf{r} \vee [\mathbf{x}]_1 = [\mathbf{A}_1]_1 \cdot \mathbf{r}\}.$$

Note that this OR-proof is implicitly given in [GOS12, Ràf15]. For the sake of completeness we recall the proof here.

Lemma 41. *If the $\mathcal{D}_k$-MDDH assumption holds in the group $\mathbb{G}_2$, then the proof system $\mathsf{PS} := (\mathsf{PGen}, \mathsf{PTGen}, \mathsf{PPrv}, \mathsf{PVer}, \mathsf{PSim})$ as defined in Figure 4.1 is a non-interactive zero-knowledge proof for $\mathcal{L}^{\mathrm{snd}}$. More precisely, for every adversary $\mathcal{A}$ attacking the composable zero-knowledge property of PS, we obtain an adversary $\mathcal{B}$ with $T(\mathcal{B}) \approx T(\mathcal{A}) + Q_{\mathrm{prove}} \cdot \mathsf{poly}(\lambda)$ and*

$$\mathrm{Adv}^{\mathrm{zk}}_{\mathsf{PS},\mathcal{A}}(\lambda) \leq \mathrm{Adv}^{\mathrm{mddh}}_{\mathcal{PG},\mathbb{G}_2,\mathcal{D}_k,\mathcal{B}}(\lambda).$$

Proof. **Completeness:** Let $j \in \{0,1\}$ such that $[\mathbf{x}]_1 = [\mathbf{A}_j]_1 \cdot \mathbf{r}$. Further, let $([\mathbf{z}_0]_2, ([\mathbf{C}_i]_2, [\Pi_i]_1)_{i\in\{0,1\}})$ be returned by PPrv on input crs, $[\mathbf{x}]_1$ and $\mathbf{r}$.

$$\begin{aligned} e([\mathbf{A}_j]_1, [\mathbf{C}_j]_2) &= e([\mathbf{A}_j]_1, \mathbf{S}_j \cdot [\mathbf{D}]_2^\top + \mathbf{r} \cdot [\mathbf{z}_j]_2^\top) = [\mathbf{A}_j \cdot \mathbf{S}_j \cdot \mathbf{D}^\top]_T + [\mathbf{A}_j \cdot \mathbf{r} \cdot \mathbf{z}_j^\top]_T \\ &= [\Pi_j \cdot \mathbf{D}^\top]_T + [\mathbf{x} \cdot \mathbf{z}_j^\top]_T = e([\Pi_j]_1, [\mathbf{D}]_2^\top) + e([\mathbf{x}]_1, [\mathbf{z}_j]_2^\top) \end{aligned}$$

and further

$$\begin{aligned} e([\mathbf{A}_{1-j}]_1, [\mathbf{C}_{1-j}]_2) &= e([\mathbf{A}_{1-j}]_1, \mathbf{S}_{1-j} \cdot [\mathbf{D}]_2^\top) = [\mathbf{A}_{1-j} \cdot \mathbf{S}_{1-j} \cdot \mathbf{D}^\top]_T \\ &= [(\mathbf{A}_{1-j} \cdot \mathbf{S}_{1-j} - \mathbf{x} \cdot \mathbf{v}^\top + \mathbf{x} \cdot \mathbf{v}^\top) \cdot \mathbf{D}^\top]_T \\ &= [\Pi_{1-j} \cdot \mathbf{D}^\top]_T + [\mathbf{x} \cdot \mathbf{z}_{1-j}^\top]_T \\ &= e([\Pi_{1-j}]_1, [\mathbf{D}]_2^\top) + e([\mathbf{x}]_1, [\mathbf{z}_{1-j}]_2^\top). \end{aligned}$$

Composable zero-knowledge: Let $\mathcal{A}$ be a PPT adversary, attacking the zero-knowledge property. We build a PPT adversary $\mathcal{B}$ such that $T(\mathcal{B}) \approx T(\mathcal{A})$ and

$$\mathrm{Adv}^{\mathrm{zk}}_{\mathsf{PS},\mathcal{A}}(\lambda) \leq \mathrm{Adv}^{\mathrm{mddh}}_{\mathcal{PG},\mathbb{G}_2,\mathcal{D}_k,\mathcal{B}}(\lambda) + \frac{1}{p}.$$

Upon receiving its MDDH challenge $(\mathcal{PG}, [\mathbf{D}]_2, [\mathbf{z}]_2)$, $\mathcal{B}$ samples $\mathbf{A}_0, \mathbf{A}_1 \leftarrow_R \mathcal{D}_{2k,k}$ and forwards the common reference string $\mathsf{crs} := ((\mathcal{PG}, [\mathbf{A}_0]_1, [\mathbf{A}_1]_1), [\mathbf{D}]_2, [\mathbf{z}]_2)$ to $\mathcal{A}$. When $\mathcal{B}$ receives a real MDDH tuple, that is, when there exists $\mathbf{u} \in \mathbb{Z}_p^k$ such that $[\mathbf{z}]_2 := [\mathbf{D}\mathbf{u}]_2$, $\mathcal{B}$ simulates a crs as output by $\mathsf{PTGen}(1^\lambda, \mathsf{pars})$. The other case is when $\mathcal{B}$ receives $[\mathbf{z}]_2 \leftarrow_R \mathbb{G}_2^{k+1}$. In that case, using the fact that the uniformly random distribution over $\mathbb{Z}_p^{k+1}$ and the uniformly random distribution over $\mathbb{Z}_p^{k+1} \setminus \mathrm{span}(\mathbf{D})$ are $1/p$-statistically close distributions, since $\mathbf{D}$ is of rank k, we can conclude that $\mathcal{B}$ simulates the crs as output by $\mathsf{PGen}(1^\lambda, \mathsf{pars})$, within a $1/p$ statistical distance. Overall, we get: $\mathrm{Adv}^{\mathrm{zk}}_{\mathsf{PS},\mathcal{A}}(\lambda) \leq \mathrm{Adv}^{\mathrm{mddh}}_{\mathcal{PG},\mathbb{G}_2,\mathcal{D}_k,\mathcal{B}}(\lambda) + \frac{1}{p}$.

Now, we proceed to prove that for all $[\mathbf{x}]_1 \in \mathcal{L}^{\text{snd}}$ with witness $\mathbf{r} \in \mathbb{Z}_p^k$,

$$\{\mathsf{PPrv}(\mathsf{crs}, [\mathbf{x}]_1, \mathbf{r}), (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{PTGen}(1^\lambda, \mathsf{pars})\}$$

is identically distributed to

$$\{\mathsf{PSim}(\mathsf{crs}, \mathsf{td}, [\mathbf{x}]_1), (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{PTGen}(1^\lambda, \mathsf{pars})\},$$

which concludes the proof.

First, note that PPrv and PSim compute the vectors $[\mathbf{z}_0]_2$ and $[\mathbf{z}_1]_2$ in the exact same way, i.e. for all $j \in \{0,1\}$, $\mathbf{z}_j := \mathbf{D}\mathbf{v}_j$ where $\mathbf{v}_0, \mathbf{v}_1$ are uniformly random over $\mathbb{Z}_p^k$ subject to $\mathbf{v}_0 + \mathbf{v}_1 = \mathbf{u}$ (recall $\mathbf{z} := \mathbf{D}\mathbf{u}$). Second, on input $[\mathbf{x}]_1 := [\mathbf{A}_j\mathbf{r}]_1$, for some $j \in \{0,1\}$, $\mathsf{PPrv}(\mathsf{crs}, [\mathbf{x}]_1, \mathbf{r})$ computes $[\mathbf{C}_{1-j}]_2$ and $[\Pi_{1-j}]_1$ exactly as PSim, that is: $[\mathbf{C}_{1-j}]_2 = \mathbf{S}_{1-j}[\mathbf{D}^\top]_2$ and $[\Pi_{1-j}]_1 = [\mathbf{A}_{1-j}]_1\mathbf{S}_{1-j} - [\mathbf{x}]_1 \cdot \mathbf{v}_{1-j}^\top$. The algorithm $\mathsf{PPrv}(\mathsf{crs}, [\mathbf{x}]_1, \mathbf{r})$ additionally computes $[\mathbf{C}_j]_2 = \mathbf{S}_j[\mathbf{D}^\top]_2 + \mathbf{r} \cdot [\mathbf{z}_j^\top]_2$ and $[\Pi_j]_1 = [\mathbf{A}_j]_1\mathbf{S}_j$, with $\mathbf{S}_j \leftarrow_R \mathbb{Z}_p^{k\times k}$. Since the following are identically distributed:

$$\mathbf{S}_j \quad \text{and} \quad \mathbf{S}_j - \mathbf{r} \cdot \mathbf{v}_j^\top,$$

for $\mathbf{S}_j \leftarrow_R \mathbb{Z}_p^{k\times k}$, we can re-write the commitment and proof computed by $\mathsf{PPrv}(\mathsf{crs}, [\mathbf{x}]_1, \mathbf{r})$ as $[\mathbf{C}_j]_2 = \mathbf{S}_j[\mathbf{D}^\top]_2 - \mathbf{r} \cdot \mathbf{v}_j^\top[\mathbf{D}^\top]_2 + \mathbf{r} \cdot [\mathbf{z}_j^\top]_2 = [\mathbf{S}_j\mathbf{D}^\top]_2$ and $[\Pi_j]_1 = [\mathbf{A}_j]_1\mathbf{S}_j - [\mathbf{A}_j\mathbf{r} \cdot \mathbf{v}_j^\top\mathbf{D}^\top]_2 = [\mathbf{A}_j\mathbf{S}_j]_1 - [\mathbf{x} \cdot \mathbf{z}_j^\top]_2$, which is exactly as the output of PSim.

Perfect soundness: Since $\mathbf{z} = \mathbf{z}_0 + \mathbf{z}_1 \notin \text{span}(\mathbf{D})$, there is a $j \in \{0,1\}$ such that $\mathbf{z}_j \notin \text{span}(\mathbf{D})$. This implies that there exists a $\mathbf{d}^\perp \in \mathbb{Z}_p^{k+1}$ such that $\mathbf{D}^\top\mathbf{d}^\perp = \mathbf{0}$, and $\mathbf{z}_j^\top\mathbf{d}^\perp = 1$. Furthermore, as the row vectors of $\mathbf{D}$ together with $\mathbf{z}_j$ form a basis of $\mathbb{Z}_p^{k+1}$, we can write $[\mathbf{C}_j]_2 := [\mathbf{S}_j \cdot \mathbf{D}^\top + \mathbf{r} \cdot \mathbf{z}_j^\top]_2$ for some $\mathbf{S}_j \in \mathbb{Z}_p^{k\times k}, \mathbf{r} \in \mathbb{Z}_p^k$. Multiplying the verification equation by $\mathbf{d}^\perp$ thus yields $[\mathbf{A}_j\mathbf{r}]_1 = [\mathbf{x}]_1$, which proves a successful forgery outside $\mathcal{L}^{\text{snd}}$ impossible.

□

4.2 Our Core Lemma

We start this work by providing our so-called core lemma, which captures the essential randomization technique from [AHN+17, Hof17, GHK17]. We can employ this lemma to prove the security of our MAC and (structure-preserving) signature schemes. Essentially, the core lemma shows that the term $[\mathbf{k}_0^\top\mathbf{t}]_1$ is pseudorandom. We give the corresponding formal experiment in Figure 4.2.

Lemma 42 (Core lemma)**.** *If the $\mathcal{D}_{2k,k}$-MDDH assumption holds in $\mathbb{G}_1$ and the tuple of algorithms $\mathsf{PS} := (\mathsf{PGen}, \mathsf{PTGen}, \mathsf{PPrv}, \mathsf{PVer})$ is a non-interactive zero-knowledge proof system for $\mathcal{L}^{\text{snd}}$, then going from experiment $\text{Exp}^{\text{core}}_{0,\mathcal{A}}(\lambda)$ to $\text{Exp}^{\text{core}}_{1,\mathcal{A}}(\lambda)$ can (up to negligible terms) only increase the winning chances of an adversary. More precisely, for every adversary $\mathcal{A}$, there exist adversaries $\mathcal{B}$, $\mathcal{B}'$ with running time $T(\mathcal{B}) \approx T(\mathcal{B}') \approx T(\mathcal{A}) + Q \cdot \mathsf{poly}(\lambda)$ such that*

$$\text{Adv}^{\text{core}}_{0,\mathcal{A}}(\lambda) \leq \text{Adv}^{\text{core}}_{1,\mathcal{A}}(\lambda) + \Delta^{\text{core}}_{\mathcal{A}}(\lambda),$$

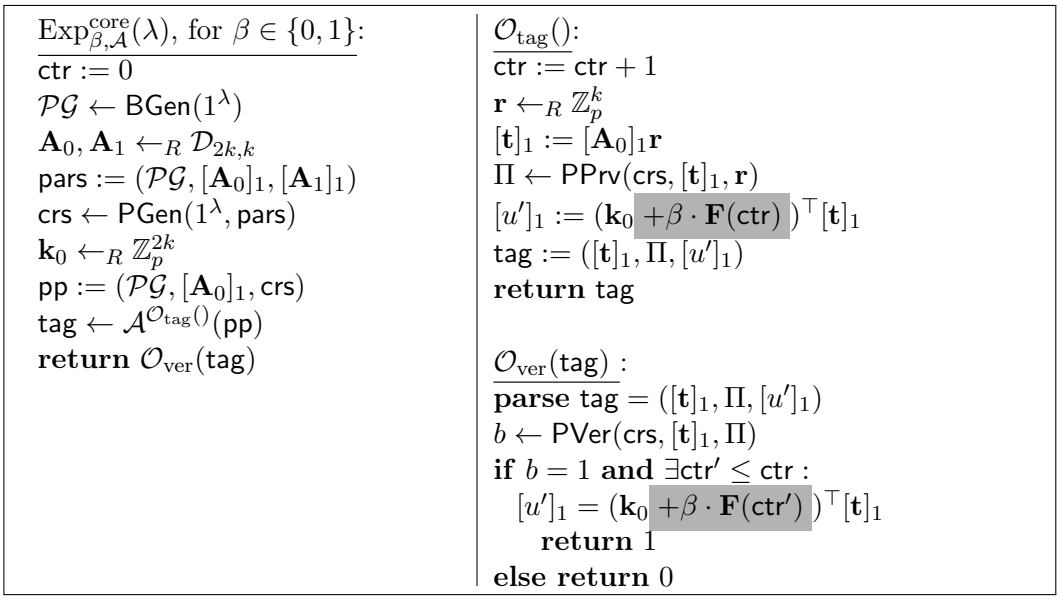

Figure 4.2: Experiment for the core lemma. Here, $\mathbf{F} : \mathbb{Z}_p \to \mathbb{Z}_p^{2k}$ is a random function computed on the fly. We highlight the difference between $\mathrm{Exp}^{\mathrm{core}}_{0,\mathcal{A}}$ and $\mathrm{Exp}^{\mathrm{core}}_{1,\mathcal{A}}$ in gray.

where

$$\begin{aligned}\Delta^{\mathrm{core}}_{\mathcal{A}}(\lambda) :=& (4k\lceil \log Q \rceil + 2) \cdot \mathrm{Adv}^{\mathrm{mddh}}_{\mathcal{PG},\mathbb{G}_1,\mathcal{D}_{2k,k},\mathcal{B}}(\lambda) \\ &+ (2\lceil \log Q \rceil + 2) \cdot \mathrm{Adv}^{\mathrm{ZK}}_{\mathsf{PS},\mathcal{B}'}(\lambda) \\ &+ \lceil \log Q \rceil \cdot \Delta_{\mathcal{D}_{2k,k}} + \frac{4\lceil \log Q \rceil + 2}{p-1} + \frac{\lceil \log Q \rceil \cdot Q}{p}.\end{aligned}$$

Recall that by definition of the distribution $\mathcal{D}_{2k,k}$ (Section 2.5), the term $\Delta_{\mathcal{D}_{2k,k}}$ is statistically small.

Proof outline. Since the proof of Theorem 42 is rather complex, we first outline our strategy. Intuitively, our goal is to randomize the term u' used by oracles $\mathcal{O}_{\mathrm{tag}}$ and $\mathcal{O}_{\mathrm{ver}}$ (i.e., to change this term from $\mathbf{k}_0^\top \mathbf{t}$ to $(\mathbf{k}_0 + \mathbf{F}(\mathsf{ctr}))^\top \mathbf{t}$ for a truly random function $\mathbf{F}$). In this, it will also be helpful to change the distribution of $\mathbf{t} \in \mathbb{Z}_p^{2k}$ in tags handed out by $\mathcal{O}_{\mathrm{tag}}$ as needed. (Intuitively, changing $\mathbf{t}$ can be justified with the $\mathcal{D}_{2k,k}$-MDDH assumption, but we can only rely on the soundness of PS if $\mathbf{t} \in \mathrm{span}(\mathbf{A}_0) \cup \mathrm{span}(\mathbf{A}_1)$. In other words, we may assume that $\mathbf{t} \in \mathrm{span}(\mathbf{A}_0) \cup \mathrm{span}(\mathbf{A}_1)$ for any of $\mathcal{A}$'s $\mathcal{O}_{\mathrm{ver}}$ queries, but only if the same holds for all $\mathbf{t}$ chosen by $\mathcal{O}_{\mathrm{tag}}$.)

We will change u' using a hybrid argument, where in the i-th hybrid we set $u' = (\mathbf{k}_0^\top + \mathbf{F}_i(\mathsf{ctr}_{|i}))^\top \mathbf{t}$ for a random function $\mathbf{F}_i$ on i-bit prefixes, and the i-bit prefix $\mathsf{ctr}_{|i}$ of ctr. (That is, we introduce more and more dependencies on the bits of ctr.) To move from hybrid i to hybrid $i+1$, we proceed again along a series of hybrids (outsourced into the proof of Theorem 43),and perform the following modifications:

Partitioning. First, we choose $\mathbf{t} \in \mathrm{span}(\mathbf{A}_{\mathsf{ctr}_{i+1}})$ in $\mathcal{O}_{\mathrm{ver}}$, where ctr_{i+1} is the $(i+1)$-th bit of ctr. As noted above, this change can be justified with the $\mathcal{D}_{2k,k}$-MDDH assumption, and we may still assume $\mathbf{t} \in \mathrm{span}(\mathbf{A}_0) \cup \mathrm{span}(\mathbf{A}_1)$ in every $\mathcal{O}_{\mathrm{tag}}$ query from $\mathcal{A}$.

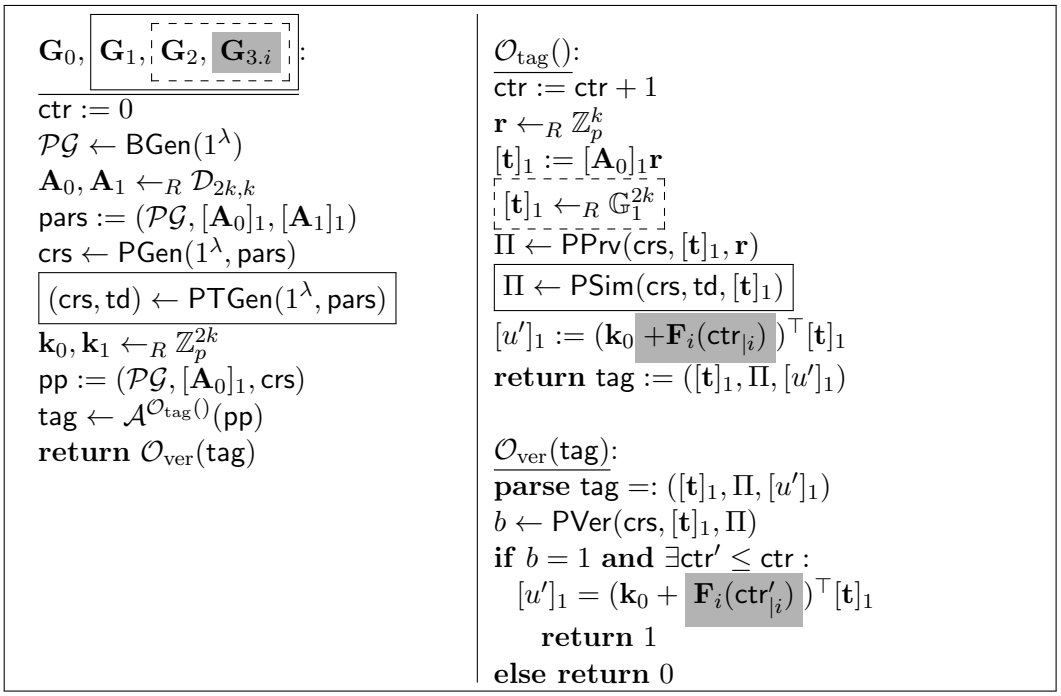

Figure 4.3: Games $\mathbf{G}_0, \mathbf{G}_1, \mathbf{G}_2, \mathbf{G}_{3.i}$ for $i \in \{0, \ldots, \lceil \log Q \rceil - 1\}$, for the proof of the core lemma (Lemma 42). $\mathbf{F}_i : \{0,1\}^i \to \mathbb{Z}_p^{2k}$ denotes a random function, and $\mathsf{ctr}_{|i}$ denotes the i-bit prefix of the counter ctr written in binary. In each procedure, the components inside a solid (dotted, gray) frame are only present in the games marked by a solid (dotted, gray) frame.

Decoupling. At this point, the values u' computed in $\mathcal{O}_{\mathrm{tag}}$ and $\mathcal{O}_{\mathrm{ver}}$ are either of the form $u' = (\mathbf{k}_0^\top + \mathbf{F}_i(\mathsf{ctr}_{|i}))^\top \mathbf{A}_0 \mathbf{r}$ or $u' = (\mathbf{k}_0^\top + \mathbf{F}_i(\mathsf{ctr}_{|i}))^\top \mathbf{A}_1 \mathbf{r}$ (depending on $\mathbf{t}$). Since $\mathbf{F}_i : \{0,1\}^i \to \mathbb{Z}_p^{2k}$ is truly random, and the matrix $\mathbf{A}_0 || \mathbf{A}_1 \in \mathbb{Z}_p^{2k \times 2k}$ has linearly independent columns (with overwhelming probability), the two possible subterms $\mathbf{F}_i(\mathsf{ctr}_{|i})^\top \mathbf{A}_0$ and $\mathbf{F}_i(\mathsf{ctr}_{|i})^\top \mathbf{A}_1$ are independent. Thus, switching to $u' = (\mathbf{k}_0^\top + \mathbf{F}_{i+1}(\mathsf{ctr}_{|i+1}))^\top \mathbf{t}$ does not change $\mathcal{A}$'s view at all.

After these modifications (and resetting $\mathbf{t}$), we have arrived at the $(i+1)$-th hybrid, which completes the proof. However, this outline neglects a number of details, including a proper reasoning of PS proofs, and a careful discussion of the decoupling step. In particular, an additional complication arises in this step from the fact that an adversary may choose $\mathbf{t} \in \mathrm{span}(A_b)$ for an arbitrary bit b not related to any specific ctr. This difficulty is the reason for the somewhat surprising "$\exists \mathsf{ctr}' \leq \mathsf{ctr}$" clause in $\mathcal{O}_{\mathrm{ver}}$.

Proof of Theorem 42. We proceed via a series of hybrid games $\mathbf{G}_0, \ldots, \mathbf{G}_{3.\lceil \log Q \rceil}$, described in Figure 4.3, and we denote by ε_i the advantage of $\mathcal{A}$ to win $\mathbf{G}_i$, that is $\Pr[\mathbf{G}_i(\mathcal{A}, 1^\lambda) = 1]$, where the probability is taken over the random coins of $\mathbf{G}_i$ and $\mathcal{A}$.

Game $\mathbf{G}_0$: We have $\mathbf{G}_0 = \mathrm{Exp}^{\mathrm{core}}_{0,\mathcal{A}}(\lambda)$ and thus by definition:

$$\varepsilon_0 = \mathrm{Adv}^{\mathrm{core}}_{0,\mathcal{A}}(\lambda).$$

Transition $\mathbf{G}_0 \rightsquigarrow \mathbf{G}_1$: Game $\mathbf{G}_1$ is as $\mathbf{G}_0$, except that crs is generated by PTGen instead of PGen. Because the output of PSim and PPrv are identically distributed on a $\mathsf{crs} \leftarrow_R \mathsf{PTGen}$ (see Theorem 40), we can argue that the crs distribution is the only difference in these two games. This difference is justified by the zero-knowledge of PS. Namely, we build an adversary $\mathcal{B}$ on the composable zero-knowledge property of PS as follows. The adversary $\mathcal{B}$ obtains crs from its own experiment instead of calling PGen, samples $\mathbf{A}_0 \leftarrow_R \mathcal{D}_{2k,k}$, and forwards $\mathsf{pars} := (\mathcal{PG}, [\mathbf{A}_0]_1, \mathsf{crs})$ to $\mathcal{A}$. Then $\mathcal{B}$ samples $\mathbf{k}_0, \mathbf{k}_1 \leftarrow_R \mathbb{Z}_p^{2k}$, thanks to which it can answer $\mathcal{O}_{\text{tag}}$ and $\mathcal{O}_{\text{ver}}$ queries. Note that $\mathcal{B}$ simulates $\mathbf{G}_0$ in case it was given a crs generated by PGen, whereas it simulates $\mathbf{G}_1$ in case it was given a crs generated by PTGen. Thus, $\mathcal{B}$ is such that $T(\mathcal{B}) \approx T(\mathcal{A}) + Q \cdot \mathsf{poly}(\lambda)$ and

$$|\varepsilon_0 - \varepsilon_1| \leq \mathrm{Adv}^{\mathrm{ZK}}_{\mathsf{PS},\mathcal{B}}(\lambda).$$

Transition $\mathbf{G}_1 \rightsquigarrow \mathbf{G}_2$: We can switch $[\mathbf{t}]_1$ to random over $\mathbb{G}_1$ by applying the $\mathcal{D}_{2k,k}$ assumption. More precisely, let $\mathcal{A}$ be an adversary distinguishing between $\mathbf{G}_1$ and $\mathbf{G}_2$ and let $\mathcal{B}$ be an adversary given a Q-fold $\mathcal{D}_{2k,k}$-MDDH challenge $(\mathcal{PG}, [\mathbf{A}_0]_1, [\mathbf{z}_1]_1, \ldots, [\mathbf{z}_Q]_1)$ as input. Now $\mathcal{B}$ sets up the game for $\mathcal{A}$ similar to $\mathbf{G}_1$, but instead choosing $\mathbf{A}_0 \leftarrow_R \mathcal{D}_{2k,k}$, it uses its challenge matrix $[\mathbf{A}_0]_1$ as part of the public parameters pars. Further, to answer tag queries $\mathcal{B}$ sets $[\mathbf{t}_i]_1 := [\mathbf{z}_i]_1$ and computes the rest accordingly. This is possible as the proof Π is simulated from game $\mathbf{G}_1$ on. In case $\mathcal{B}$ was given a real $\mathcal{D}_{2k,k}$-challenge, it simulates $\mathbf{G}_1$ and otherwise $\mathbf{G}_2$. Lemma 3 yields the existence of an adversary $\mathcal{B}_1$ with $T(\mathcal{B}_1) \approx T(\mathcal{A}) + Q \cdot \mathsf{poly}(\lambda)$ and

$$|\varepsilon_1 - \varepsilon_2| \leq k \cdot \mathrm{Adv}^{\mathrm{mddh}}_{\mathcal{PG},\mathbb{G}_1,\mathcal{D}_{2k,k},\mathcal{B}_1}(\lambda) + \frac{1}{p-1}.$$

Transition $\mathbf{G}_2 \rightsquigarrow \mathbf{G}_{3.0}$: As for all $\mathsf{ctr} \in \mathbb{N}$ we have $\mathbf{F}_0(\mathsf{ctr}_{|0}) = \mathbf{F}_0(\epsilon)$ and $\mathbf{k}_0$ is distributed identically to $\mathbf{k}_0 + \mathbf{F}_0(\epsilon)$ for $\mathbf{k}_0 \leftarrow_R \mathbb{Z}_p^{2k}$ we have

$$\varepsilon_2 = \varepsilon_{3.0}.$$

Transition $\mathbf{G}_{3.i} \rightsquigarrow \mathbf{G}_{3.(i+1)}$: For the proof of this transition we refer to Lemma 43, which states that for every adversary $\mathcal{A}$ there exist adversaries $\mathcal{B}_i$, $\mathcal{B}'_i$ with running time $T(\mathcal{B}_i) \approx T(\mathcal{B}'_i) \approx T(\mathcal{A}) + Q \cdot \mathsf{poly}(\lambda)$, and

$$\begin{aligned}\varepsilon_{3.i} \leq & \varepsilon_{3.(i+1)} + 4k \cdot \mathrm{Adv}^{\mathrm{mddh}}_{\mathcal{PG},\mathbb{G}_1,\mathcal{D}_{2k,k},\mathcal{B}_i}(\lambda) + 2\mathrm{Adv}^{\mathrm{ZK}}_{\mathsf{PS},\mathcal{B}'_i}(\lambda) \\ & + \Delta_{\mathcal{D}_{2k,k}} + \frac{4}{p-1} + \frac{Q}{p}.\end{aligned}$$

Transition $\mathbf{G}_{3.\lceil \log Q \rceil} \rightsquigarrow \mathrm{Exp}^{\mathrm{core}}_{1,\mathcal{A}}(\lambda)$: It is left to reverse the changes introduced in the transitions from game $\mathbf{G}_0$ to game $\mathbf{G}_2$ to end up at the experiment $\mathrm{Exp}^{\mathrm{core}}_{1,\mathcal{A}}(1^\lambda)$. In order to do so we introduce an intermediary game $\mathbf{G}_4$, where we set $[\mathbf{t}] := [\mathbf{A}_0]_1\mathbf{r}$ for $\mathbf{r} \leftarrow_R \mathbb{Z}_p^k$. This corresponds to reversing transition $\mathbf{G}_1 \rightsquigarrow \mathbf{G}_2$. By the same reasoning for every adversary $\mathcal{A}$ we thus obtain an adversary $\mathcal{B}_{3.\lceil \log Q \rceil}$ with $T(\mathcal{B}_{3.\lceil \log Q \rceil}) \approx T(\mathcal{A}) + Q \cdot \mathsf{poly}(\lambda)$ such that

$$|\varepsilon_{3.\lceil \log Q \rceil} - \varepsilon_4| \leq k \cdot \mathrm{Adv}^{\mathrm{mddh}}_{\mathcal{PG},\mathbb{G}_1,\mathcal{D}_{2k,k},\mathcal{B}_{3.\lceil \log Q \rceil}}(\lambda) + \frac{1}{p-1}.$$

As $[\mathbf{t}]_1$ is now chosen from $\mathrm{span}([\mathbf{A}_0]_1)$ again, we can switch back to honest generation of the common reference string crs. As in transition $\mathbf{G}_0 \rightsquigarrow \mathbf{G}_1$ for an adversary $\mathcal{A}$ we obtain an adversary $\mathcal{B}_4$ with $T(\mathcal{B}_4) \approx T(\mathcal{A}) + Q \cdot \mathsf{poly}(\lambda)$ and

$$|\varepsilon_4 - \mathrm{Adv}^{\mathrm{core}}_{1,\mathcal{A}}(\lambda)| \leq \mathrm{Adv}^{\mathrm{ZK}}_{\mathsf{PS},\mathcal{B}_4}(\lambda).$$

□

Lemma 43 ($\mathbf{G}_{3.i} \rightsquigarrow \mathbf{G}_{3.(i+1)}$)**.** *If the $\mathcal{D}_{2k,k}$-MDDH assumptions holds in $\mathbb{G}_1$, and the tuple $\mathsf{PS} := (\mathsf{PGen}, \mathsf{PTGen}, \mathsf{PPrv}, \mathsf{PVer})$ is a non-interactive zero-knowledge proof system for $\mathcal{L}^{\mathrm{snd}}$, then for all $i \in \{0, \ldots, \lceil \log Q \rceil - 1\}$ between $\mathbf{G}_{3.i}$ and $\mathbf{G}_{3.(i+1)}$ as defined in Figure 4.7 the winning chances of an adversary can only increase (up to negligible terms). More precisely, for every adversary $\mathcal{A}$ there exist adversaries $\mathcal{B}_i$, $\mathcal{B}'_i$ with running times $T(\mathcal{B}_i) \approx T(\mathcal{B}'_i) \approx T(\mathcal{A}) + Q \cdot \mathsf{poly}(\lambda)$, and*

$$\begin{aligned}\varepsilon_{3.i} \leq & \varepsilon_{3.(i+1)} + 4k \cdot \mathrm{Adv}^{\mathrm{mddh}}_{\mathcal{PG},\mathbb{G}_1,\mathcal{D}_{2k,k},\mathcal{B}_i}(\lambda) + 2\mathrm{Adv}^{\mathrm{ZK}}_{\mathsf{PS},\mathcal{B}'_i}(\lambda) \\ & + \Delta_{\mathcal{D}_{2k,k}} + \frac{4}{p-1} + \frac{Q}{p}.\end{aligned}$$

Proof. We proceed via a series of hybrid games $\mathbf{H}_{i.j}$ for $i \in \{0, \ldots, \lceil \log Q \rceil - 1\}$, $j \in \{1, \ldots, 8\}$, described in Figure 4.4, and we denote by $\widehat{\varepsilon}_{i.j}$ the advantage of $\mathcal{A}$ to win $\mathbf{H}_{i.j}$. We give an overview of the transitions in Figure 4.5.

Transition $\mathbf{G}_{3.i} \rightsquigarrow \mathbf{H}_{i.1}$: We switch $[\mathbf{t}]_1$ from chosen uniformly at random by $\mathcal{O}_{\mathrm{tag}}$ to $[\mathbf{A}_{\mathsf{ctr}_{i+1}}\mathbf{r}]_1$ for $\mathbf{r} \leftarrow_R \mathbb{Z}_p^k$, where ctr_{i+1} is the $i+1$'st bit of the binary representation of ctr, using the $\mathcal{D}_{2k,k}$-MDDH assumption twice. More precisely, we introduce an intermediary game $\mathbf{H}_{i.0}$, where we choose $[\mathbf{t}_i]_1$ as

$$[\mathbf{t}_i]_1 = \begin{cases} [\mathbf{A}_0\mathbf{r}_i] \text{ for } \mathbf{r}_i \leftarrow_R \mathbb{Z}_p^k & \text{if } \mathsf{ctr}_{i+1} = 0 \\ [\mathbf{u}_i]_1 \text{ for } \mathbf{u}_i \leftarrow_R \mathbb{Z}_p^{2k} & \text{else} \end{cases}.$$

Let $\mathcal{A}$ be an adversary distinguishing between $\mathbf{G}_{3.i}$ and $\mathbf{H}_{i.0}$ and let $\mathcal{B}$ be an adversary receiving a Q-fold MDDH-challenge $(\mathcal{PG}, [\mathbf{A}_0]_1, [\mathbf{z}_1]_1, \ldots, [\mathbf{z}_Q]_1)$ as input. Then $\mathcal{B}$ sets up the game for $\mathcal{A}$ similar to game $\mathbf{G}_{3.i}$, where he embeds $[\mathbf{A}_0]_1$ into the public parameters pars. Further, whenever obtaining a simulation query ctr with $\mathsf{ctr}_{|i+1} = 0$, $\mathcal{B}$ sets $[\mathbf{t}_i] := [\mathbf{z}_i]_1$ and otherwise follows $\mathbf{G}_{3.i}$. Similar, we can reduce the transition from game $\mathbf{H}_{i.0}$ to $\mathbf{H}_{i.1}$ to the $\mathcal{D}_{2k,k}$-MDDH assumption. Applying Lemma 3 yields an adversary $\mathcal{B}_{i.0}$ with $T(\mathcal{B}_{i.0}) \approx T(\mathcal{A}) + Q \cdot \mathsf{poly}(\lambda)$ such that:

$$|\varepsilon_{3.i} - \widehat{\varepsilon}_{i.1}| \leq 2k \cdot \mathrm{Adv}^{\mathrm{mddh}}_{\mathcal{PG},\mathbb{G}_1,\mathcal{D}_{2k,k},\mathcal{B}_{i.0}}(\lambda) + \frac{2}{p-1}.$$

Transition $\mathbf{H}_{i.1} \rightsquigarrow \mathbf{H}_{i.2}$: In this step we reverse the transition from game $\mathbf{G}_0$ to $\mathbf{G}_1$ in Theorem 44. Namely, we generate crs using PGen instead of PTGen, and we use the fact that proofs generated by PSim or PPrv are identically distributed when $\mathsf{crs} \leftarrow_R \mathsf{PTGen}(1^\lambda, \mathsf{pars})$. Note that it is possible to use the algorithm PPrv, as from game $\mathbf{H}_{i.1}$ on, we choose all $[\mathbf{t}]_1$ in tag queries from $\mathcal{L}$ with corresponding witness and can thus honestly generate proofs.

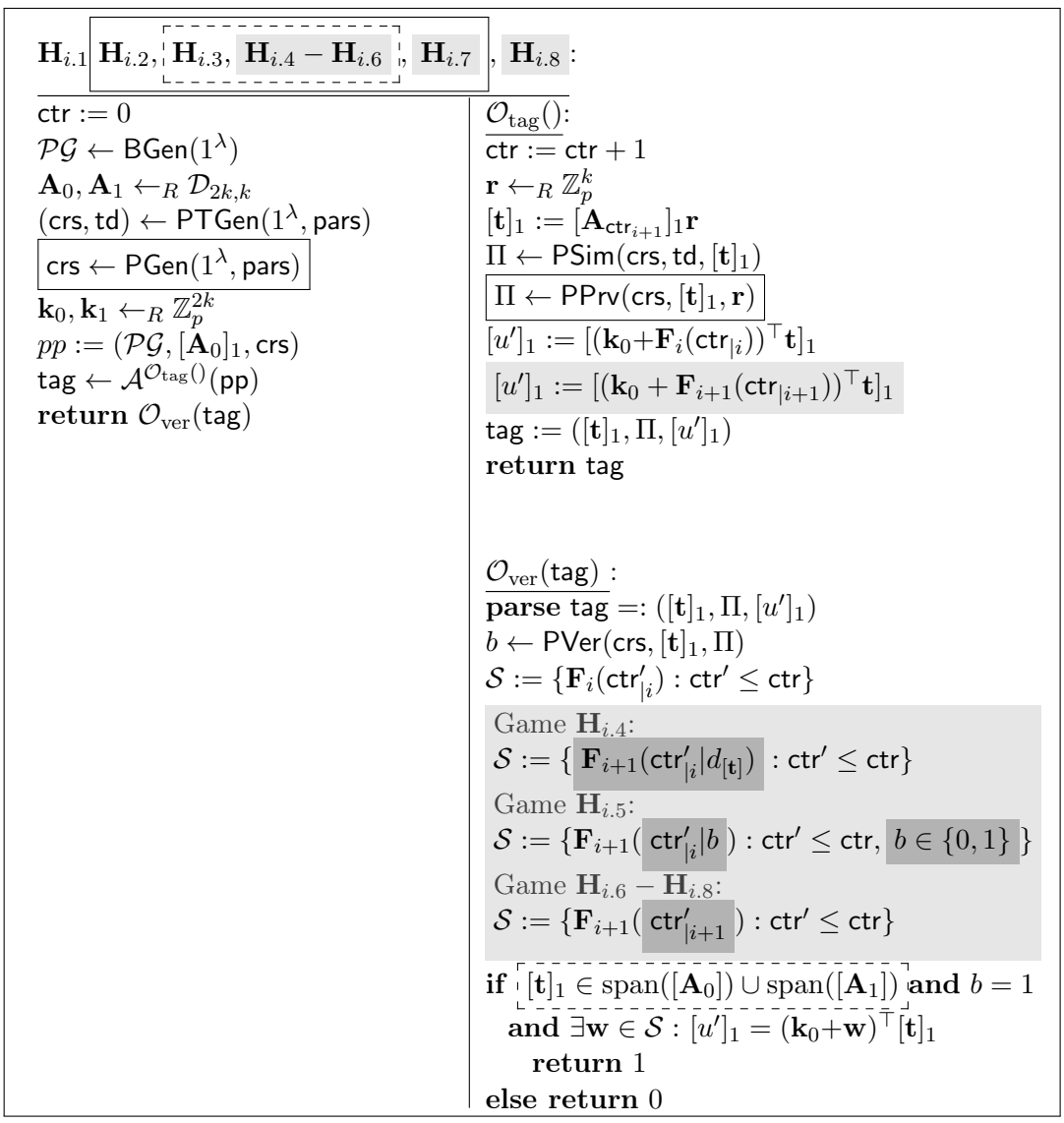

Figure 4.4: Games $\mathbf{H}_{i.j}$ for $i \in \{0, \ldots, \lceil \log Q \rceil - 1\}$, $j \in \{1, \ldots, 8\}$, for the proof of Lemma 43. Here, $\mathbf{F}_i : \{0,1\}^i \to \mathbb{Z}_p^{2k}$ denotes a random function, $\mathsf{ctr}_{|i}$ denotes the i-bit string that is a prefix of ctr written in binary, and ctr_i is the i'th bit of ctr written in binary. We have $d_{[\mathbf{t}]} = 0$ if $\mathbf{t} \in \mathrm{span}(\mathbf{A}_0)$, and $d_{[\mathbf{t}]} = 1$ if $\mathbf{t} \in \mathrm{span}(\mathbf{A}_1)$. In each procedure, the components inside a solid (dotted, gray) frame are only present in the games marked by a solid (dotted, gray) frame. For the intermediate transitions from game $\mathbf{H}_{i.4}$ to game $\mathbf{H}_{i.6}$ we use dark gray highlighting to emphasize the respective differences.

Therefore, by the same reasoning as for $\mathbf{G}_0 \rightsquigarrow \mathbf{G}_1$ there exists an adversary $\mathcal{B}_{i.1}$ such that $T(\mathcal{B}_{i.1}) \approx T(\mathcal{A}) + Q \cdot \mathsf{poly}(\lambda)$ and

$$|\widehat{\varepsilon}_{i.1} - \widehat{\varepsilon}_{i.2}| \leq \mathrm{Adv}^{\mathrm{ZK}}_{\mathsf{PS},\mathcal{B}_{i.1}}(\lambda).$$

Transition $\mathbf{H}_{i.2} \rightsquigarrow \mathbf{H}_{i.3}$: From game $\mathbf{H}_{i.3}$ on we introduce an additionally check in the verification oracle. Namely, $\mathcal{O}_{\mathrm{ver}}$ checks that $[\mathbf{t}]_1 \in \mathrm{span}([\mathbf{A}_0]_1) \cup$

$\mathrm{span}([\mathbf{A}_1]_1)$. As the crs is generated by PGen, we can employ the perfect soundness of PS to obtain

$$\widehat{\varepsilon}_{i.2} = \widehat{\varepsilon}_{i.3}.$$

Transition $\mathbf{H}_{i.3} \rightsquigarrow \mathbf{H}_{i.4}$: Let $\mathbf{A}_0^\perp \in \mathrm{orth}(\mathbf{A}_0)$ and $\mathbf{A}_1^\perp \in \mathrm{orth}(\mathbf{A}_1)$. We introduce an intermediary game $\mathbf{H}_{i.3.1}$, where we replace the random function $\mathbf{F}_i \colon \{0,1\}^i \to \mathbb{Z}_p^{2k}$ by

$$\mathbf{F}'_i \colon \{0,1\}^i \to \mathbb{Z}_p^{2k}, \;\; \mathbf{F}'_i(\nu) := \left(\mathbf{A}_0^\perp | \mathbf{A}_1^\perp\right) \begin{pmatrix} \Gamma_i(\nu) \\ \Upsilon_i(\nu) \end{pmatrix},$$

where $\nu \in \{0,1\}^i$ is a i-bit string and $\Gamma_i, \Upsilon_i : \{0,1\}^i \to \mathbb{Z}_p^k$ are two independent random functions. With probability $1 - \Delta_{\mathcal{D}_{2k,k}}$ the matrix $(\mathbf{A}_0^\perp | \mathbf{A}_1^\perp)$ has full rank. In this case, going from game $\mathbf{H}_{i.3}$ to game $\mathbf{H}_{i.3.1}$ consists merely in a change of basis, thus, these two games are perfectly indistinguishable. We obtain $|\widehat{\varepsilon}_{3.i} - \widehat{\varepsilon}_{3.i.1}| \leq \Delta_{\mathcal{D}_{2k,k}}$.

We now define

$$\mathbf{F}_{i+1} \colon \{0,1\}^{i+1} \to \mathbb{Z}_p^{2k}, \;\; \mathbf{F}_{i+1}(\nu) := \begin{cases} \left(\mathbf{A}_0^\perp | \mathbf{A}_1^\perp\right) \begin{pmatrix} \Gamma'_i(\nu_{|i}) \\ \Upsilon_i(\nu_{|i}) \end{pmatrix} & \text{if } \nu_{i+1} = 0 \\ \left(\mathbf{A}_0^\perp | \mathbf{A}_1^\perp\right) \begin{pmatrix} \Gamma_i(\nu_{|i}) \\ \Upsilon'_i(\nu_{|i}) \end{pmatrix} & \text{else} \end{cases},$$

where $\Gamma'_i, \Upsilon'_i : \{0,1\}^i \to \mathbb{Z}_p^k$ are fresh independent random functions. Now $\mathbf{F}_{i+1}$ constitutes a random function $\{0,1\}^{i+1} \to \mathbb{Z}_p^{2k}$.

Replacing $\mathbf{F}'_i(\mathsf{ctr}_{|i})$ by $\mathbf{F}_{i+1}(\mathsf{ctr}_{|i+1})$ does not show up in any of the tag queries, as we have

$$\begin{aligned} \mathbf{F}_{i+1}(\mathsf{ctr}_{i+1})^\top [\mathbf{t}]_1 &= \mathbf{F}_{i+1}(\mathsf{ctr}_{i+1})^\top [\mathbf{A}_{\mathsf{ctr}_{i+1}}]_1 \mathbf{r} \\ &= \begin{cases} \Gamma'_i(\mathsf{ctr}_{|i}) \mathbf{A}_0^\perp [\mathbf{A}_0]_1 \mathbf{r} + \Upsilon_i(\mathsf{ctr}_{|i}) \mathbf{A}_1^\perp [\mathbf{A}_0]_1 \mathbf{r} & \text{if } \mathsf{ctr}_{i+1} = 0 \\ \Gamma_i(\mathsf{ctr}_{|i}) \mathbf{A}_0^\perp [\mathbf{A}_1]_1 \mathbf{r} + \Upsilon'_i(\mathsf{ctr}_{|i}) \mathbf{A}_1^\perp [\mathbf{A}_1]_1 \mathbf{r} & \text{else} \end{cases} \\ &= \begin{cases} \Upsilon_i(\mathsf{ctr}_{|i}) \mathbf{A}_1^\perp [\mathbf{A}_0]_1 \mathbf{r} & \text{if } \mathsf{ctr}_{i+1} = 0 \\ \Gamma_i(\mathsf{ctr}_{|i}) \mathbf{A}_0^\perp [\mathbf{A}_1]_1 \mathbf{r} & \text{else} \end{cases} \\ &= \mathbf{F}'_i(\mathsf{ctr}_{|i})^\top [\mathbf{A}_{\mathsf{ctr}_{i+1}}]_1 \mathbf{r}. \end{aligned}$$

In the verification oracle we check $[\mathbf{t}]_1 \in \mathrm{span}([\mathbf{A}_0]) \cup \mathrm{span}([\mathbf{A}_1])$, define $d_{[\mathbf{t}]} = 0$ if $\mathbf{t} \in \mathrm{span}(\mathbf{A}_0)$ and $d_{[\mathbf{t}]} = 1$ if $\mathbf{t} \in \mathrm{span}(\mathbf{A}_1)$ and replace $\mathbf{F}_i(\mathsf{ctr}_{|i})$ by $\mathbf{F}_{i+1}(\mathsf{ctr}_{|i} | d_{[\mathbf{t}]})$. Thus, by similar reasoning as for tag queries, the change does not show up in the final verification query either.

Altogether, we obtain

$$|\widehat{\varepsilon}_{3.3} - \widehat{\varepsilon}_{3.4}| \leq \Delta_{\mathcal{D}_{2k,k}}.$$

Transition $\mathbf{H}_{i.4} \rightsquigarrow \mathbf{H}_{i.5}$: From game $\mathbf{H}_{i.5}$ on, we extend the set $\mathcal{S}$ in the verification oracle from $\mathcal{S}_{i.4} := \{\mathbf{F}_{i+1}(\mathsf{ctr}'_{|i} | d_{[\mathbf{t}]}) : \mathsf{ctr}' \leq \mathsf{ctr}\}$ to $\mathcal{S}_{i.5} := \{\mathbf{F}_{i+1}(\mathsf{ctr}'_{|i} | b) : \mathsf{ctr}' \leq \mathsf{ctr}, b \in \{0,1\}\}$. That is, we regard a verification query $([\mathbf{t}]_1, \Pi, [u']_1)$ as valid, if

#	$\mathsf{crs} \leftarrow$	$[\mathbf{t}]_1$ in $\mathcal{O}_{\text{tag}}$	$\Pi \leftarrow$ in $\mathcal{O}_{\text{tag}}$	$[u']_1 = (\mathbf{k}_0 + \cdot)^\top [\mathbf{t}]_1$ in $\mathcal{O}_{\text{tag}}$	$\mathcal{S} := \{\cdot \colon \mathsf{ctr}' \in \mathcal{Q}_{\text{tag}}\}$ in $\mathcal{O}_{\text{ver}}$	check on $[\mathbf{t}]_1$ in $\mathcal{O}_{\text{ver}}$	**game knows**	**remark**
$\mathbf{G}_{3.i}$	PTGen	$\leftarrow_R \mathbb{G}_1^{2k}$	PSim	$\mathbf{F}_i(\mathsf{ctr}_{\lvert i})$	$\mathbf{F}_i(\mathsf{ctr}'_{\lvert i})$	-	-	Game $\mathbf{G}_{3.i}$
$\mathbf{H}_{i.1}$	PTGen	$= [\mathbf{A}_{\mathsf{ctr}_{i+1}}]_1 \mathbf{r}$	PSim	$\mathbf{F}_i(\mathsf{ctr}_{\lvert i})$	$\mathbf{F}_i(\mathsf{ctr}'_{\lvert i})$	-	-	$\mathcal{D}_{2k,k}$-MDDH
$\mathbf{H}_{i.2}$	PGen	$= [\mathbf{A}_{\mathsf{ctr}_{i+1}}]_1 \mathbf{r}$	PPrv	$\mathbf{F}_i(\mathsf{ctr}_{\lvert i})$	$\mathbf{F}_i(\mathsf{ctr}'_{\lvert i})$	-	-	ZK of PS
$\mathbf{H}_{i.3}$	PGen	$= [\mathbf{A}_{\mathsf{ctr}_{i+1}}]_1 \mathbf{r}$	PPrv	$\mathbf{F}_i(\mathsf{ctr}_{\lvert i})$	$\mathbf{F}_i(\mathsf{ctr}'_{\lvert i})$	$[\mathbf{t}]_1 \overset{?}{\in} \mathcal{L}^{\text{snd}}$	$\mathbf{A}_0, \mathbf{A}_1$	SND of PS
$\mathbf{H}_{i.4}$	PGen	$= [\mathbf{A}_{\mathsf{ctr}_{i+1}}]_1 \mathbf{r}$	PPrv	$\mathbf{F}_{i+1}(\mathsf{ctr}_{\lvert i+1})$	$\mathbf{F}_{i+1}(\mathsf{ctr}'_{\lvert i} \lvert d_{[\mathbf{t}]})$	$[\mathbf{t}]_1 \overset{?}{\in} \mathcal{L}^{\text{snd}}$	$\mathbf{A}_0, \mathbf{A}_1$	statistical
$\mathbf{H}_{i.5}$	PGen	$= [\mathbf{A}_{\mathsf{ctr}_{i+1}}]_1 \mathbf{r}$	PPrv	$\mathbf{F}_{i+1}(\mathsf{ctr}_{\lvert i+1})$	$\mathbf{F}_{i+1}(\mathsf{ctr}'_{\lvert i} \lvert b), b \in \{0,1\}$	$[\mathbf{t}]_1 \overset{?}{\in} \mathcal{L}^{\text{snd}}$	$\mathbf{A}_0, \mathbf{A}_1$	incr. chances
$\mathbf{H}_{i.6}$	PGen	$= [\mathbf{A}_{\mathsf{ctr}_{i+1}}]_1 \mathbf{r}$	PPrv	$\mathbf{F}_{i+1}(\mathsf{ctr}_{\lvert i+1})$	$\mathbf{F}_{i+1}(\mathsf{ctr}'_{\lvert i+1})$	$[\mathbf{t}]_1 \overset{?}{\in} \mathcal{L}^{\text{snd}}$	$\mathbf{A}_0, \mathbf{A}_1$	statistical
$\mathbf{H}_{i.7}$	PGen	$= [\mathbf{A}_{\mathsf{ctr}_{i+1}}]_1 \mathbf{r}$	PPrv	$\mathbf{F}_{i+1}(\mathsf{ctr}_{\lvert i+1})$	$\mathbf{F}_{i+1}(\mathsf{ctr}'_{\lvert i+1})$	-	-	SND of PS
$\mathbf{H}_{i.8}$	PTGen	$= [\mathbf{A}_{\mathsf{ctr}_{i+1}}]_1 \mathbf{r}$	PSim	$\mathbf{F}_{i+1}(\mathsf{ctr}_{\lvert i+1})$	$\mathbf{F}_{i+1}(\mathsf{ctr}'_{\lvert i+1})$	-	-	ZK of PS
$\mathbf{G}_{3.(i+1)}$	PTGen	$\leftarrow_R \mathbb{G}_1^{2k}$	PSim	$\mathbf{F}_{i+1}(\mathsf{ctr}_{\lvert i+1})$	$\mathbf{F}_{i+1}(\mathsf{ctr}'_{\lvert i+1})$	-	-	$\mathcal{D}_{2k,k}$-MDDH

Figure 4.5: Overview of the transitions in the proof of Lemma 43. We highlight the respective changes between the games in gray. In the third column, $\mathbf{r}$ is chosen at random from $\mathbb{Z}_p^k$ and ctr_{i+1} denotes the $i+1$'st bit of the bit representation of $\mathsf{ctr} \in \mathbb{Z}_p$. In the fifth and sixth column, the dot $\cdot$ represents the gap filled by the respective entries in the table. Further, $\mathbf{F}_i \colon \{0,1\}^i \to \mathbb{Z}_p^{2k}$, $\mathbf{F}_{i+1} \colon \{0,1\}^i \to \mathbb{Z}_p^{2k}$ are random functions and $\mathsf{ctr}_{\lvert i}$ and $\mathsf{ctr}_{\lvert i+1}$ denote the bit strings consisting of the first i respectively the first $i+1$ bits of the bit representation of $\mathsf{ctr} \in \mathbb{N}$. Further, we have $d_{[\mathbf{t}]} = 0$ if $\mathbf{t} \in \text{span}(\mathbf{A}_0)$, and $d_{[\mathbf{t}]} = 1$ if $\mathbf{t} \in \text{span}(\mathbf{A}_1)$. For the seventh column, recall that $\mathcal{L}^{\text{snd}} := \text{span}([\mathbf{A}_0]_1) \cup \text{span}([\mathbf{A}_1]_1)$. In the remark we give the justification for the respective transition from the previous game to the current game.

there exists a $\mathsf{ctr}' \leq \mathsf{ctr}$ such that $[u']_1 = (\mathbf{k}_0 + \mathbf{F}_{i+1}(\mathsf{ctr}'_{|i}|b))^\top [\mathbf{t}]_1$ for $b \in \{0,1\}$ arbitrary, instead of requiring $b = d_{[\mathbf{t}]}$ (where $d_{[\mathbf{t}]} = 0$ if $\mathbf{t} \in \mathrm{span}(\mathbf{A}_0)$ and $d_{[\mathbf{t}]} = 1$ if $\mathbf{t} \in \mathrm{span}(\mathbf{A}_1)$). As changing the verification oracle does not change the view of the adversary before providing its output and as we have $\mathcal{S}_{i.4} \subseteq \mathcal{S}_{i.5}$, the transition from game $\mathbf{H}_{i.4}$ to game $\mathbf{H}_{i.5}$ can only increase the chance of the adversary. We thus have

$$\widehat{\varepsilon}_{i.4} \leq \widehat{\varepsilon}_{i.5}.$$

Transition $\mathbf{H}_{i.5} \rightsquigarrow \mathbf{H}_{i.6}$: The difference between game $\mathbf{H}_{i.5}$ and game $\mathbf{H}_{i.6}$ is that in the latter we only regard a verification query $([\mathbf{t}]_1, \Pi, [u]_1)$ valid, if there exists a $\mathsf{ctr}' \leq \mathsf{ctr}$ such that $[u]_1 = (\mathbf{k}_0 + \mathbf{F}_{i+1}(\mathsf{ctr}'_{|i}|\mathsf{ctr}'_{i+1}))^\top [\mathbf{t}]_1$ (instead of allowing the last bit to be arbitrary). As the only way an adversary can learn the image of $\mathbf{F}_{i+1}$ on a value is via tag queries and $\mathbf{F}_{i+1}$ is a random function, a union bound over the elements in $\mathcal{Q}_{\mathrm{tag}}$ yields

$$|\widehat{\varepsilon}_{i.5} - \widehat{\varepsilon}_{i.6}| \leq \frac{Q}{p}.$$

Transition $\mathbf{H}_{i.6} \rightsquigarrow \mathbf{H}_{i.7}$: The oracle $\mathcal{O}_{\mathrm{ver}}$ does not perform the additional check $[\mathbf{t}]_1 \in \mathrm{span}([\mathbf{A}_0]_1) \cup \mathrm{span}([\mathbf{A}_1]_1)$ anymore from game $\mathbf{H}_{i.7}$ on. This is justified by the soundness of PS. As in transition $\mathbf{H}_{i.2} \rightsquigarrow \mathbf{H}_{i.3}$ we obtain

$$\widehat{\varepsilon}_{i.6} = \widehat{\varepsilon}_{i.7}.$$

Transition $\mathbf{H}_{i.7} \rightsquigarrow \mathbf{H}_{i.8}$: This transition is similar to transition $\mathbf{G}_0$ to $\mathbf{G}_1$ in Theorem 44. Namely, for an adversary $\mathcal{A}$ distinguishing the two games, we can employ the composable zero-knowledge property of PS to obtain an adversary $\mathcal{B}_{i.7}$ such that $T(\mathcal{B}_{i.8}) \approx T(\mathcal{A}) + Q \cdot \mathsf{poly}(\lambda)$ and

$$|\widehat{\varepsilon}_{i.7} - \widehat{\varepsilon}_{i.8}| \leq \mathrm{Adv}^{\mathrm{ZK}}_{\mathsf{PS},\mathcal{B}_{i.7}}(\lambda).$$

Transition $\mathbf{H}_{i.8} \rightsquigarrow \mathbf{G}_{3.(i+1)}$: We switch $[\mathbf{t}]_1$ generated by $\mathcal{O}_{\mathrm{tag}}$ to uniformly random over $\mathbb{G}_1^{2k}$, using the $\mathcal{D}_{2k,k}$-MDDH assumption first on $[\mathbf{A}_0]_1$, then on $[\mathbf{A}_1]_1$. Similarly than for the transition $\mathbf{G}_{3.i} \rightsquigarrow \mathbf{H}_{i.1}$, we obtain an adversary $\mathcal{B}_{i.8}$ with $T(\mathcal{B}_{i.8}) \approx T(\mathcal{A}) + Q \cdot \mathsf{poly}(\lambda)$ such that

$$|\widehat{\varepsilon}_{i.8} - \varepsilon_{3.(i+1)}| \leq 2k \cdot \mathrm{Adv}^{\mathrm{mddh}}_{\mathcal{PG},\mathbb{G}_1,\mathcal{D}_{2k,k},\mathcal{B}_{i.8}}(\lambda) + \frac{2}{p-1}.$$

□

4.3 Our Tightly Secure Message Authentication Code Scheme

Let $k \in \mathbb{N}$ and let $\mathsf{PS} := (\mathsf{PGen}, \mathsf{PTGen}, \mathsf{PPrv}, \mathsf{PSim})$ a non-interactive zero-knowledge proof for $\mathcal{L}^{\mathrm{snd}}$ as defined in Section 4.1. In Figure 4.6 we provide a MAC $\mathsf{MAC} := (\mathsf{Gen}, \mathsf{Tag}, \mathsf{Ver})$ whose security can be tightly reduced to $\mathcal{D}_{2k,k}$-MDDH and the security of the underlying proof system PS.

$\underline{\mathsf{Gen}(1^\lambda)}$:
$\mathcal{PG} \leftarrow \mathsf{BGen}(1^\lambda)$
$\mathbf{A}_0, \mathbf{A}_1 \leftarrow \mathcal{D}_{2k,k}$
$\mathsf{pars} := (\mathcal{PG}, [\mathbf{A}_0]_1, [\mathbf{A}_1]_1)$
$\mathsf{crs} \leftarrow \mathsf{PGen}(1^\lambda, \mathsf{pars})$
$\mathbf{k}_0, \mathbf{k}_1 \leftarrow_R \mathbb{Z}_p^{2k}$
$\mathsf{pp} := (\mathcal{PG}, [\mathbf{A}_0]_1, \mathsf{crs})$
$\mathsf{sk} := (\mathbf{k}_0, \mathbf{k}_1)$
return (pp, sk)

$\underline{\mathsf{Tag}(\mathsf{pp}, \mathsf{sk}, \mu \in \mathbb{Z}_p)}$:
parse $\mathsf{pp} =: (\mathcal{PG}, [\mathbf{A}_0]_1, \mathsf{crs})$
$\mathbf{r} \leftarrow_R \mathbb{Z}_p^k$
$[\mathbf{t}]_1 := [\mathbf{A}_0]_1 \mathbf{r}$
$\Pi \leftarrow \mathsf{PPrv}(\mathsf{crs}, [\mathbf{t}]_1, \mathbf{r})$
$[u]_1 := (\mathbf{k}_0 + \mu \mathbf{k}_1)^\top [\mathbf{t}]_1$
$\mathsf{tag} := ([\mathbf{t}]_1, \Pi, [u]_1)$
return tag

$\underline{\mathsf{Ver}(\mathsf{pp}, \mathsf{sk}, \mu \in \mathbb{Z}_p, \mathsf{tag})}$:
parse $\mathsf{tag} =: ([\mathbf{t}]_1, \Pi, [u]_1)$
$b \leftarrow \mathsf{PVer}(\mathsf{crs}, [\mathbf{t}]_1, \Pi)$
if $b = 1$ **and** $[u]_1 \neq [0]_1$
and $[u]_1 = (\mathbf{k}_0 + \mu \mathbf{k}_1)^\top [\mathbf{t}]_1$
return 1
else return 0

Figure 4.6: Tightly secure MAC $\mathsf{MAC} := (\mathsf{Gen}, \mathsf{Tag}, \mathsf{Ver})$ from the $\mathcal{D}_{2k,k}$-MDDH assumption.

Theorem 44 (EUF-CMA security of MAC)**.** *If the $\mathcal{D}_{2k,k}$-MDDH assumptions holds in $\mathbb{G}_1$, and the tuple $\mathsf{PS} := (\mathsf{PGen}, \mathsf{PTGen}, \mathsf{PPrv}, \mathsf{PVer})$ is a non-interactive zero-knowledge proof system for $\mathcal{L}^{\mathrm{snd}}$, then the MAC $\mathsf{MAC} := (\mathsf{Gen}, \mathsf{Tag}, \mathsf{Ver})$ provided in Figure 4.6 is EUF-CMA secure. Namely, for any adversary $\mathcal{A}$, there exists an adversary $\mathcal{B}$ with running time $T(\mathcal{B}) \approx T(\mathcal{A}) + Q \cdot \mathsf{poly}(\lambda)$, where Q is the number of queries to $\mathcal{O}_{\mathrm{tag}}$, poly is independent of Q, and*

$$\mathrm{Adv}^{\text{euf-cma}}_{\mathsf{MAC},\mathcal{A}}(\lambda) \leq \Delta^{\mathrm{core}}_{\mathcal{B}}(\lambda) + \frac{Q}{p}.$$

Proof. We prove the claim via a series of games, described in Figure 4.7. For $i \in \{0, 1\}$, by ε_i we denote the advantage of $\mathcal{A}$ to win game $\mathbf{G}_i$, that is $\Pr[\mathbf{G}_i(\mathcal{A}, 1^\lambda) = 1]$, where the probability is taken over the random coins of $\mathbf{G}_i$ and $\mathcal{A}$.

Game $\mathbf{G}_0$: We define game $\mathbf{G}_0$ to be the EUF-CMA security game $\mathrm{Exp}^{\text{euf-cma}}_{\mathcal{A}}(\lambda)$ and have thus

$$\varepsilon_0 = \mathrm{Adv}^{\text{euf-cma}}_{\mathsf{MAC},\mathcal{A}}(\lambda).$$

Transition $\mathbf{G}_0 \rightsquigarrow \mathbf{G}_1$: Let $\mathcal{A}$ be an adversary distinguishing between $\mathbf{G}_0$ and $\mathbf{G}_1$. Then we construct an adversary $\mathcal{B}$ with $T(\mathcal{B}) \approx T(\mathcal{A}) + Q \cdot \mathsf{poly}(\lambda)$ allowing to break the core lemma (Lemma 42) as follows. On input pp from $\mathrm{Exp}^{\mathrm{core}}_{\beta}(1^\lambda, \mathcal{B})$ the adversary $\mathcal{B}$ forwards pp to $\mathcal{A}$. Then, $\mathcal{B}$ samples $\mathbf{k}_1 \leftarrow_R \mathbb{Z}_p^{2k}$. Afterwards, on a tag query μ from $\mathcal{A}$, $\mathcal{B}$ queries its own $\mathcal{O}_{\mathrm{tag}}$ oracle (which takes no input), receives $([\mathbf{t}]_1, \Pi, [u']_1)$, computes $[u]_1 := [u']_1 + \mu \mathbf{k}_1^\top [\mathbf{t}]_1$, and answers with $([\mathbf{t}]_1, \Pi, [u]_1)$. Finally, given the forgery $\big(\mu^\star, \mathsf{tag}^\star := ([\mathbf{t}]_1, \Pi, [u^\star]_1)\big)$ from $\mathcal{A}$, if $\mu^\star \notin \mathcal{Q}_{\mathrm{tag}}$ and $[u^\star]_1 \neq [0]_1$, then the adversary $\mathcal{B}$ sends $\mathsf{tag}' := ([\mathbf{t}]_1, \Pi, [u^\star]_1 + \mu \mathbf{k}_1^\top [\mathbf{t}]_1)$ to its experiment (otherwise an invalid tuple). Then we have $\varepsilon_0 = \mathrm{Adv}^{\mathrm{core}}_{0,\mathcal{B}}(\lambda)$ and $\varepsilon_1 = \mathrm{Adv}^{\mathrm{core}}_{1,\mathcal{B}}(\lambda)$. The core lemma yields

$$\mathrm{Adv}^{\mathrm{core}}_{0,\mathcal{B}}(\lambda) \leq \mathrm{Adv}^{\mathrm{core}}_{1,\mathcal{B}}(\lambda) + \Delta^{\mathrm{core}}_{\mathcal{B}}(\lambda)$$

$\mathbf{G}_0, \boxed{\mathbf{G}_1}$:
$\mathcal{Q}_{\text{tag}} := \emptyset$
$\boxed{\mathsf{ctr} := 0}$
$\mathcal{PG} \leftarrow \mathsf{BGen}(1^\lambda)$
$\mathbf{A}_0, \mathbf{A}_1 \leftarrow_R \mathcal{D}_{2k,k}$
$\mathsf{pars} := (\mathcal{PG}, [\mathbf{A}_0]_1, [\mathbf{A}_1]_1)$
$\mathsf{crs} \leftarrow \mathsf{PGen}(1^\lambda, \mathsf{pars})$
$\mathbf{k}_0, \mathbf{k}_1 \leftarrow_R \mathbb{Z}_p^{2k}$
$\mathsf{pp} := (\mathcal{PG}, [\mathbf{A}_0]_1, \mathsf{crs})$
$(\mu^\star, \mathsf{tag}^\star) \leftarrow \mathcal{A}^{\mathcal{O}_{\text{tag}}(\cdot)}(\mathsf{pp})$
if $\mu^\star \notin \mathcal{Q}_{\text{tag}}$
  and $\mathcal{O}_{\text{ver}}(\mu^\star, \mathsf{tag}^\star) = 1$
  return 1
else return 0

$\mathcal{O}_{\text{tag}}(\mu)$:
$\mathcal{Q}_{\text{tag}} := \mathcal{Q}_{\text{tag}} \cup \{\mu\}$
$\boxed{\mathsf{ctr} := \mathsf{ctr} + 1}$
$\mathbf{r} \leftarrow_R \mathbb{Z}_p^k$
$[\mathbf{t}]_1 := [\mathbf{A}_0]_1 \mathbf{r}$
$\Pi \leftarrow \mathsf{PPrv}(\mathsf{crs}, [\mathbf{t}]_1, \mathbf{r})$
$[u]_1 := (\mathbf{k}_0 + \mu \mathbf{k}_1 \boxed{+\mathbf{F}(\mathsf{ctr})})^\top [\mathbf{t}]_1$
$\mathsf{tag} := ([\mathbf{t}]_1, \Pi, [u]_1)$
return tag

$\mathcal{O}_{\text{ver}}(\mu^\star, \mathsf{tag}^\star)$:
parse $\mathsf{tag}^\star =: ([\mathbf{t}]_1, \Pi, [u]_1)$
$b \leftarrow \mathsf{PVer}([\mathbf{t}]_1, \Pi)$
if $b = 1$ **and** $[u]_1 \neq [0]_1$ **and** $\boxed{\exists \mathsf{ctr}' \leq \mathsf{ctr}:}$
  $[u]_1 = (\mathbf{k}_0 + \mu^\star \mathbf{k}_1 \boxed{+\mathbf{F}_i(\mathsf{ctr}')})^\top [\mathbf{t}]_1$
  return 1
else return 0

Figure 4.7: Games $\mathbf{G}_0$ (corresponding to the EUF-CMA security experiment) and game $\mathbf{G}_1$ for the EUF-CMA proof of MAC in Figure 4.6. $\mathbf{F} : \{0,1\}^{\lceil \log Q \rceil} \to \mathbb{Z}_p^{2k}$ denotes a random function, applied on ctr written in binary. The instructions inside a solid frame are only present in game $\mathbf{G}_1$.

and thus we obtain

$$\varepsilon_0 - \varepsilon_1 \leq \Delta_{\mathcal{B}}^{\text{core}}(\lambda).$$

Game $\mathbf{G}_1$: We now prove that any adversary $\mathcal{A}$ has only negligible chances to win game $\mathbf{G}_1$ using the randomness of $\mathbf{F}$ together with the pairwise independence of $\mu \mapsto \mathbf{k}_0 + \mu \mathbf{k}_1$.

Let $(\mu^\star, \mathsf{tag}^\star)$ be the forgery of $\mathcal{A}$. we can replace $\mathbf{k}_1$ by $\mathbf{k}_1 - \mathbf{v}$ for $\mathbf{v} \leftarrow_R \mathbb{Z}_p^{2k}$, as both are distributed identically. Next, for all $j \leq Q$ we can replace $\mathbf{F}(j)$ by $\mathbf{F}(j) + \mu^{(j)} \cdot \mathbf{v}$ for the same reason. This way, $\mathcal{O}_{\text{tag}}(\mu^{(j)})$ computes

$$\begin{aligned}[u^{(j)}]_1 :&= [(\mathbf{k}_0 + \mu^{(j)} \mathbf{k}_1 \boxed{-\mu^{(j)} \mathbf{v}} + \mathbf{F}(j) \boxed{+\mu^{(j)} \mathbf{v}})^\top \mathbf{t}^{(j)}]_1 \\ &= [(\mathbf{k}_0 + \mu^{(j)} \mathbf{k}_1 + \mathbf{F}(j)^\top \mathbf{t}^{(j)}]_1,\end{aligned}$$

and $\mathcal{O}_{\text{ver}}\big([\mu^\star]_2, \mathsf{tag}^\star := ([\mathbf{t}]_1, \Pi, [u])\big)$ checks if there exists a counter $i \in \mathcal{Q}_{\text{tag}}$ such that:

$$\begin{aligned}[u]_1 &= [(\mathbf{k}_0 + \mu^\star \mathbf{k}_1 \boxed{-\mu^\star \mathbf{v}} + \mathbf{F}(i) \boxed{+\mu^{(i)} \mathbf{v}})^\top \mathbf{t}]_1 \\ &= [(\mathbf{k}_0 + \mu^\star \mathbf{k}_1 + \mathbf{F}(i)^\top \mathbf{t}^\star]_1 \boxed{+[(\mu^{(i)} - \mu^\star) \mathbf{v}^\top \mathbf{t}]_1}.\end{aligned}$$

For the forgery to be successful, it must hold $\mu^\star \notin \mathcal{Q}_{\text{tag}}$ and $[u] \neq 0$ (and thus $[\mathbf{t}]_1 \neq [\mathbf{0}]_1$). Therefore, each value computed by $\mathcal{O}_{\text{ver}}$ is (marginally) uniformly random over $\mathbb{G}_1$.

$\underline{\mathsf{Gen}(1^\lambda)}$:
$\mathcal{PG} \leftarrow \mathsf{BGen}(1^\lambda)$
$\mathbf{A}_0, \mathbf{A}_1 \leftarrow \mathcal{D}_{2k,k}$
$\mathsf{pars} := (\mathcal{PG}, [\mathbf{A}_0]_1, [\mathbf{A}_1]_1)$
$\mathsf{crs} \leftarrow \mathsf{PGen}(1^\lambda, \mathsf{pars})$
$\mathbf{A} \leftarrow_R \mathcal{D}_k$
$\mathbf{K}_0, \mathbf{K}_1 \leftarrow_R \mathbb{Z}_p^{2k\times(k+1)}$
$\mathsf{pk} := (\mathcal{PG}, [\mathbf{A}_0]_1, \mathsf{crs}, [\mathbf{A}]_2, [\mathbf{K}_0\mathbf{A}]_2, [\mathbf{K}_1\mathbf{A}]_2)$
$\mathsf{sk} := (\mathbf{K}_0, \mathbf{K}_1)$
return (pk, sk)

$\underline{\mathsf{Sign}(\mathsf{pk}, \mathsf{sk}, \mu \in \mathbb{Z}_p)}$:
$\mathbf{r} \leftarrow_R \mathbb{Z}_p^k$
$[\mathbf{t}]_1 := [\mathbf{A}_0]_1\mathbf{r}$
$\Pi \leftarrow \mathsf{PPrv}(\mathsf{crs}, [\mathbf{t}]_1, \mathbf{r})$
$[\mathbf{u}]_1 := (\mathbf{K}_0 + \mu\mathbf{K}_1)^\top [\mathbf{t}]_1$
$\sigma := ([\mathbf{t}]_1, \Pi, [\mathbf{u}]_1)$
return σ

$\underline{\mathsf{Ver}(\mathsf{pk}, \mu \in \mathbb{Z}_p, \sigma)}$:
parse $\mathsf{tag} =: ([\mathbf{t}]_1, \Pi, [\mathbf{u}]_1)$
$b \leftarrow \mathsf{PVer}(\mathsf{crs}, [\mathbf{t}]_1, \Pi)$
if $b = 1$ **and** $[\mathbf{u}]_1 \neq [\mathbf{0}]_1$ **and** $e([\mathbf{u}]_1^\top, [\mathbf{A}]_2)$
$= e([\mathbf{t}]_1^\top, [\mathbf{K}_0\mathbf{A}]_2 + \mu[\mathbf{K}_1\mathbf{A}]_2)$
return 1
else return 0

Figure 4.8: Tightly UF-CMA secure signature scheme SIG.

As the verification oracle checks for all counters $i \leq Q$, applying the union bound yields

$$\varepsilon_1 \leq \frac{Q}{p}.$$

□

4.4 Our Tightly Secure Signature Scheme

In this section, we present a signature scheme SIG for signing messages from $\mathbb{Z}_p$, described in Figure 4.8, whose UF-CMA security can be tightly reduced to the $\mathcal{D}_{2k,k}$-MDDH and $\mathcal{D}_k$-MDDH assumptions.

SIG builds upon the tightly secure MAC from Section 4.3, and functions as a stepping stone to explain the main ideas of the upcoming structure-preserving signature in Section 4.5. Recall that our MAC outputs $\mathsf{tag} = ([\mathbf{t}]_1, \Pi, [u]_1)$, where Π is a (publicly verifiable) NIZK proof of the statement $\mathbf{t} \in \mathrm{span}(\mathbf{A}_0) \cup \mathrm{span}(\mathbf{A}_1)$, and $u = (\mathbf{k}_0 + \mu\mathbf{k}_1)^\top\mathbf{t}$ has an affine structure. Hence, alternatively, we can also view our MAC as an affine MAC [BKP14] with $\mathbf{t} \in \mathrm{span}(\mathbf{A}_0) \cup \mathrm{span}(\mathbf{A}_1)$ and a NIZK proof for that. Similar to [BKP14], we use (tuned) Groth-Sahai proofs to make $[u]_1$ publicly verifiable. Similar ideas have been used to construct efficient quasi-adaptive NIZK for linear subspace [KW15, JR14], structure-preserving signatures [KPW15], and identity-based encryption schemes [BKP14].

Theorem 45 (Security of SIG)**.** *If* $\mathsf{PS} := (\mathsf{PGen}, \mathsf{PPrv}, \mathsf{PVer}, \mathsf{PSim})$ *is a non-interactive zero-knowledge proof system for* $\mathcal{L}^{\mathrm{snd}}$*, then the signature scheme* SIG *described in Figure 4.8 is* EUF-CMA *secure under the* $\mathcal{D}_{2k,k}$*-MDDH and* $\mathcal{D}_k$*-MDDH assumptions. Namely, for any adversary* $\mathcal{A}$*, there exist adversaries* $\mathcal{B}, \mathcal{B}'$ *with running time* $T(\mathcal{B}) \approx T(\mathcal{B}') \approx T(\mathcal{A}) + Q \cdot \mathsf{poly}(\lambda)$*, where* Q *is the number of queries to* $\mathcal{O}_{\mathrm{sign}}$*,* poly *is independent of* Q*, and*

$$\mathrm{Adv}^{\text{euf-cma}}_{\mathsf{SIG},\mathcal{A}}(\lambda) \leq \mathrm{Adv}^{\text{euf-cma}}_{\mathsf{MAC},\mathcal{B}}(\lambda) + \mathrm{Adv}^{\text{mddh}}_{\mathcal{PG},\mathbb{G}_2,\mathcal{D}_k,\mathcal{B}'}(\lambda).$$

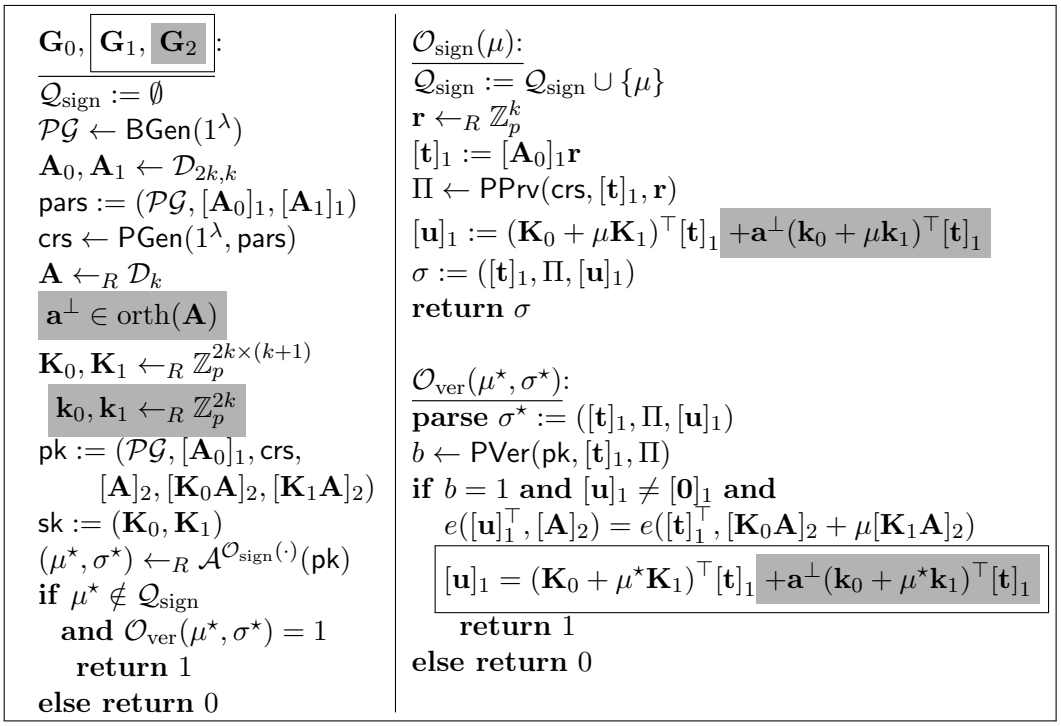

Figure 4.9: Games $\mathbf{G}_0$ (correspdonding to the EUF-CMA experiment) to $\mathbf{G}_2$ for proving Theorem 45. Note that only in $\mathbf{G}_0$ the pairing equation is used for verification. In game $\mathbf{G}_1$ und $\mathbf{G}_2$ the oracle verifies $[\mathbf{u}]_1$ directly instead (as described in the boxed instruction). The gray instructions are only added in game $\mathbf{G}_2$.

By using the KMDH assumption, we verify the forgery with the signing key; then we introduce the MAC in the kernel of $\mathbf{A}$. Since we always know $\mathbf{A}$ over $\mathbb{Z}_p$, we extract the MAC tag from the forgery and break the MAC security. The proof idea is similar, but weaker than [BKP14].

Proof. We proceed via a series of hybrid games, described in Figure 4.9. By ε_i we denote the advantage of $\mathcal{A}$ to win $\mathbf{G}_i$, that is $\Pr[\mathbf{G}_i(\mathcal{A}, 1^\lambda) = 1]$, where the probability is taken over the random coins of $\mathbf{G}_i$ and $\mathcal{A}$.

Game $\mathbf{G}_0$: We define game $\mathbf{G}_0$ to be the EUF-CMA security experiment, and thus have

$$\varepsilon_0 = \text{Exp}^{\text{euf-cma}}_{\mathsf{SIG},\mathcal{A}}(\lambda).$$

Transition $\mathbf{G}_0 \rightsquigarrow \mathbf{G}_1$: Here we change the verification oracle as described in Fig. 4.9. Note that a pair $(\mu^\star, \sigma^\star)$ that passes $\mathcal{O}_{\text{ver}}$ in $\mathbf{G}_1$ always passes the $\mathcal{O}_{\text{ver}}$ in $\mathbf{G}_0$. Thus, to bound $|\varepsilon_0 - \varepsilon_1|$, it suffices to bound the probability that $\mathcal{A}$ produces $(\mu^\star, \sigma^\star)$ that passes $\mathcal{O}_{\text{ver}}$ in $\mathbf{G}_0$ but not in $\mathbf{G}_1$. We write $\sigma^\star := ([\mathbf{t}]_1, \Pi, [\mathbf{u}]_1)$, and the verification equation in $\text{Exp}^{\text{euf-cma}}_{\mathsf{SIG},\mathcal{A}}(\lambda)$ as:

$$\begin{aligned} & e([\mathbf{u}]_1^\top, [\mathbf{A}]_2) = e([\mathbf{t}]_1^\top, [(\mathbf{K}_0 + \mu^\star\mathbf{K}_1)\mathbf{A}]_2) \\ \Leftrightarrow\ & e([\mathbf{u}]_1 - [\mathbf{t}]_1^\top(\mathbf{K}_0 + \mu^\star\mathbf{K}_1), [\mathbf{A}]_2) = \mathbf{0} \end{aligned}$$

Observe that for any $(\mu^\star, ([\mathbf{t}]_1, \Pi, [\mathbf{u}]_1))$ that passes the verification equation in $\mathbf{G}_0$ but not in $\mathbf{G}_1$ the value

$$[\mathbf{u}]_1 - [\mathbf{t}]_1^\top (\mathbf{K}_0 + \mu^\star \mathbf{K}_1)$$

is a non-zero vector in the kernel of $\mathbf{A}$.

Thus we can construct an adversary $\mathcal{B}$ with $T(\mathcal{B}) \approx T(\mathcal{A}) + Q \cdot \mathsf{poly}(\lambda)$ on the $\mathcal{D}_k$-KMDH assumption from $\mathcal{A}$ as follows. On receiving $(\mathcal{PG}, [\mathbf{A}]_2)$ from the $\mathcal{D}_k$-KMDH experiment, $\mathcal{B}$ can sample all other parameters itself and simulate $\mathbf{G}_0$ for $\mathcal{A}$. If $\mathcal{A}$ outputs the tuple $(\mu^\star, ([\mathbf{t}]_1, \Pi, [\mathbf{u}]_1))$, then $\mathcal{B}$ outputs the value $[\mathbf{u}]_1 - [\mathbf{t}]_1^\top (\mathbf{K}_0 + \mu^\star \mathbf{K}_1)$ to its own experiment.

Finally, Lemma 9 yields an adversary $\mathcal{B}'$ with $T(\mathcal{B}') \approx T(\mathcal{B}) \approx T(\mathcal{A}) + Q \cdot \mathsf{poly}(\lambda)$ and

$$|\varepsilon_0 - \varepsilon_1| \leq \mathrm{Adv}^{\mathrm{mddh}}_{\mathcal{PG},\mathbb{G}_2,\mathcal{D}_k,\mathcal{B}'}(\lambda).$$

Transition $\mathbf{G}_1 \rightsquigarrow \mathbf{G}_2$: For $i \in \{0,1\}$ we can replace $\mathbf{K}_i$ by $\mathbf{K}_i + \mathbf{k}_i(\mathbf{a}^\perp)^\top$ for $\mathbf{a}^\perp \in \mathrm{orth}(\mathbf{A})$ and $\mathbf{k}_i \leftarrow_R \mathbb{Z}_p^{2k}$, as both are distributed identically. Further, as $(\mathbf{a}^\perp)^\top \cdot \mathbf{A} = \mathbf{0}$, this change does not show up in the public key pk. Thus, we have

$$\varepsilon_1 = \varepsilon_2.$$

Transition Game $\mathbf{G}_2$: We can use the EUF-CMA security of MAC given in Figure 4.6 to bound the probability of an adversary winning game $\mathbf{G}_2$. Let $\mathcal{A}$ be an adversary on $\mathbf{G}_2$. We construct an adversary $\mathcal{B}$ with $T(\mathcal{B}) \approx T(\mathcal{A}) + Q \cdot \mathsf{poly}(\lambda)$ on the EUF-CMA security of MAC as follows.

On input $\mathsf{pp} = (\mathcal{PG}, [\mathbf{A}_0]_1, \mathsf{crs})$ of $\mathrm{Exp}^{\text{euf-cma}}_{\mathcal{A}}(\lambda)$ adversary $\mathcal{B}$ samples $\mathbf{A} \leftarrow_R \mathbf{D}_{k+1,k}$ and $\mathbf{K}_0, \mathbf{K}_1 \leftarrow_R \mathbb{Z}_p^{2k\times(k+1)}$, chooses $\mathbf{a}^\perp \in \mathrm{orth}(\mathbf{A})$ and forwards $\mathsf{pk} := (\mathcal{PG}, [\mathbf{A}_0]_1, \mathsf{crs}, [\mathbf{A}]_2, [\mathbf{K}_0\mathbf{A}]_2, [\mathbf{K}_1\mathbf{A}]_2)$ to $\mathcal{A}$.

On a signing query μ of $\mathcal{A}$, the adversary $\mathcal{B}$ queries its own tag oracle to obtain $\mathsf{tag} = ([\mathbf{t}]_1, \Pi, [u]_1)$. Then, $\mathcal{B}$ computes $[\mathbf{u}]_1 := (\mathbf{K}_0 + \mu\mathbf{K}_1)^\top [\mathbf{t}]_1 + \mathbf{a}^\perp [u]_1$ and forwards $\sigma := ([\mathbf{t}]_1, \Pi, [\mathbf{u}]_1)$ to $\mathcal{A}$.

Let $(\mu^\star, \sigma^\star)$ be a forgery of $\mathcal{A}$ with $\sigma^\star = ([\mathbf{t}^\star]_1, \Pi^\star, [\mathbf{u}^\star]_1)$. Then $\mathcal{B}$ computes $[\mathbf{u}']_1 := (\mathbf{K}_0 + \mu^\star \mathbf{K}_1)^\top [\mathbf{t}]_1$ and (if possible) chooses $[u^\star]_1$ such that $\mathbf{a}^\perp [u^\star]_1 = [\mathbf{u}^\star]_1 - [\mathbf{u}']_1$ (note that this doable efficiently given $\mathbf{a}^\perp$) Finally, $\mathcal{B}$ outputs $(\mu^\star, \mathsf{tag}^\star)$ with $\mathsf{tag}^\star := ([\mathbf{t}^\star]_1, \Pi^\star, [u^\star]_1)$. If $(\mu^\star, \sigma^\star)$ was a successful forgery of $\mathcal{A}$ then, by the definition of game $\mathbf{G}_1$, $(\mu^\star, \mathsf{tag}^\star)$ is a successful forgery in $\mathrm{Exp}^{\text{euf-cma}}_{\mathcal{A}}(\lambda)$. This yields

$$\varepsilon_2 \leq \mathrm{Adv}^{\text{euf-cma}}_{\mathsf{MAC},\mathcal{B}}(\lambda).$$

□

4.5 Our Tightly Secure Structure-Preserving Signature Scheme

In this section we present a structure-preserving signature scheme SPS, described in Figure 4.10, whose security can be tightly reduced to the $\mathcal{D}_{2k,k}$-MDDH and $\mathcal{D}_k$-MDDH assumptions. It builds upon the tightly secure signature presented in Section

$\underline{\mathsf{Gen}(1^\lambda)}$:
$\mathcal{PG} \leftarrow \mathsf{BGen}(1^\lambda)$
$\mathbf{A}_0, \mathbf{A}_1 \leftarrow_R \mathcal{D}_{2k,k}$
$\mathsf{pars} := (\mathcal{PG}, [\mathbf{A}_0]_1, [\mathbf{A}_1]_1)$
$\mathsf{crs} \leftarrow \mathsf{PGen}(\mathsf{pars}, 1^\lambda)$
$\mathbf{A} \leftarrow_R \mathcal{D}_k$
$\mathbf{K}_0 \leftarrow_R \mathbb{Z}_p^{2k\times(k+1)}$
$\mathbf{K} \leftarrow_R \mathbb{Z}_p^{(n+1)\times(k+1)}$
$\mathsf{pk} := (\mathcal{PG}, [\mathbf{A}_0]_1, \mathsf{crs}, [\mathbf{A}]_2, [\mathbf{K}_0\mathbf{A}]_2, [\mathbf{K}\mathbf{A}]_2)$
$\mathsf{sk} := (\mathbf{K}_0, \mathbf{K})$
return $(\mathsf{pk}, \mathsf{sk})$

$\underline{\mathsf{Sign}(\mathsf{pk}, \mathsf{sk}, [\mathbf{m}]_1 \in \mathbb{G}_1^n)}$:
$\mathbf{r} \leftarrow_R \mathbb{Z}_p^k$ $[\mathbf{t}]_1 := [\mathbf{A}_0]_1\mathbf{r}$
$\Pi \leftarrow \mathsf{PPrv}(\mathsf{crs}, [\mathbf{t}]_1, \mathbf{r})$
$[\mathbf{u}]_1 := \mathbf{K}_0^\top [\mathbf{t}]_1 + \mathbf{K}^\top \begin{bmatrix}\mathbf{m}\\ 1\end{bmatrix}_1$
return $\sigma := ([\mathbf{t}]_1, \Pi, [\mathbf{u}]_1)$

$\underline{\mathsf{Ver}(\mathsf{pk}, \sigma, [\mathbf{m}]_1)}$:
parse $\sigma := ([\mathbf{t}]_1, \Pi, [\mathbf{u}]_1)$
$b \leftarrow \mathsf{PVer}(\mathsf{pk}, [\mathbf{t}]_1, \Pi)$
if $b = 1$ **and** $e([\mathbf{u}]_1^\top, [\mathbf{A}]_2)$
$= e([\mathbf{t}]_1^\top, [\mathbf{K}_0\mathbf{A}]_2) + e(\begin{bmatrix}\mathbf{m}\\ 1\end{bmatrix}_1^\top, [\mathbf{K}\mathbf{A}]_2)$
return 1
else return 0

Figure 4.10: Tightly UF-CMA secure structure-preserving signature scheme SPS with message space $\mathbb{G}_1^n$.

4.4 by using a similar idea of [KPW15]. Precisely, we view μ as a label and the main difference between both schemes is that in the proof we do not need to guess which μ the adversary may reuse for its forgery, and thus our security proof is tight.

Theorem 46 (Security of SPS)**.** *If* $\mathsf{PS} := (\mathsf{PGen}, \mathsf{PTGen}, \mathsf{PVer}, \mathsf{PSim})$ *is a non-interactive zero-knowledge proof system for* $\mathcal{L}^{\mathrm{snd}}$*, the signature scheme* SPS *described in Fig. 4.10 is* EUF-CMA *secure under the* $\mathcal{D}_{2k,k}$*-MDDH and* $\mathcal{D}_k$*-MDDH assumptions. Namely, for any adversary* $\mathcal{A}$*, there exist adversaries* $\mathcal{B}, \mathcal{B}'$ *with running time* $T(\mathcal{B}) \approx T(\mathcal{B}') \approx T(\mathcal{A}) + Q \cdot \mathsf{poly}(\lambda)$*, where* Q *is the number of queries to* $\mathcal{O}_{\mathrm{sign}}$*,* poly *is independent of* Q*, and*

$$\mathrm{Adv}^{\text{euf-cma}}_{\mathsf{SPS},\mathcal{A}}(\lambda) \le \Delta^{\mathrm{core}}_{\mathcal{B}}(\lambda) + \mathrm{Adv}^{\mathrm{mddh}}_{\mathcal{PG},\mathbb{G}_2,\mathcal{D}_k,\mathcal{B}'}(\lambda) + \frac{Q}{p^k} + \frac{Q}{p}.$$

When using PS *from Section 4.1, we obtain*

$$\begin{aligned}\mathrm{Adv}^{\text{euf-cma}}_{\mathsf{SPS},\mathcal{A}}(\lambda) \le{}& (4k\lceil \log Q\rceil + 2) \cdot \mathrm{Adv}^{\mathrm{mddh}}_{\mathcal{PG},\mathbb{G}_1,\mathcal{D}_{2k,k},\mathcal{B}}(\lambda)\\ &+ (2\lceil \log Q\rceil + 3) \cdot \mathrm{Adv}^{\mathrm{mddh}}_{\mathcal{PG},\mathbb{G}_2,\mathcal{D}_k,\mathcal{B}'}(\lambda) + \lceil \log Q\rceil \cdot \Delta_{\mathcal{D}_{2k,k}}\\ &+ \frac{4\lceil \log Q\rceil + 2}{p-1} + \frac{(Q+1)\lceil \log Q\rceil + Q}{p} + \frac{Q}{p^k}.\end{aligned}$$

Strategy. In a nutshell, we will embed a "shadow MAC" in our signature scheme, and then invoke the core lemma to randomize the MAC tags computed during signing queries and the final verification of $\mathcal{A}$'s forgery. A little more specifically, we will embed a term $\mathbf{k}_0^\top\mathbf{t}$ into the $\mathbf{A}$-orthogonal space of each $\mathbf{u}$ computed by $\mathcal{O}_{\mathrm{sign}}$ and $\mathcal{O}_{\mathrm{ver}}$. (Inuitively, changes to this $\mathbf{A}$-orthogonal space do not influence the verification key, and simply correspond to changing from one signing key to another signing key that is compatible with the same verification key.) Using our core lemma, we can randomize this term $\mathbf{k}_0^\top\mathbf{t}$ to $(\mathbf{k}_0 + \mathbf{F}(\mathsf{ctr}))^\top\mathbf{t}$ for a random function $\mathbf{F}$ and a signature

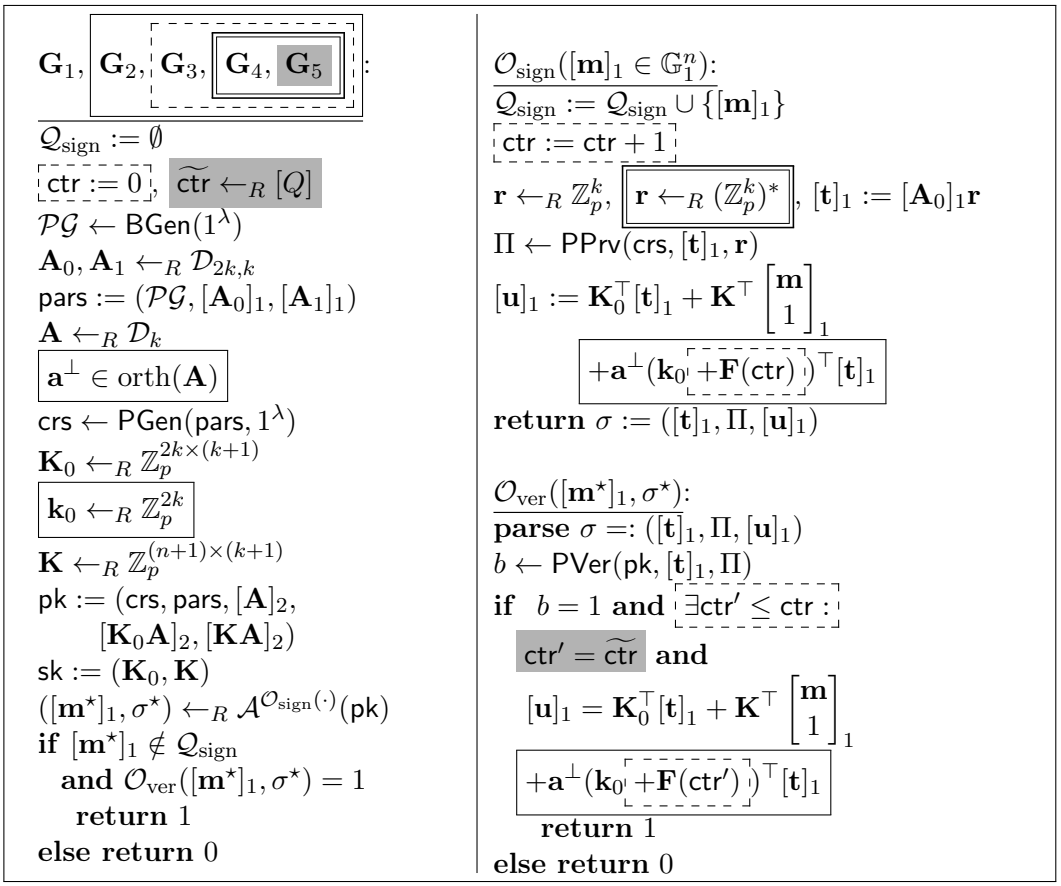

Figure 4.11: Games $\mathbf{G}_1$ to $\mathbf{G}_5$ for proving Theorem 46. Here, $\mathbf{F} : \mathbb{Z}_p \to \mathbb{Z}_p^{2k}$ is a random function. In each procedure, the components inside a solid (dotted, double, gray) frame are only present in the games marked by a solid (dotted, double, gray) frame.

counter ctr. Intuitively, this means that we use a freshly randomized signing key for each signature query. After these changes, an adversary only has a statistically small chance in producing a valid forgery.

Proof of Theorem 46. We proceed via a series of hybrid games $\mathbf{G}_0$ to $\mathbf{G}_4$, described in Figure 4.11. By ε_i we denote the advantage of $\mathcal{A}$ to win $\mathbf{G}_i$.

Game $\mathbf{G}_0$: We define game $\mathbf{G}_0$ to be the EUF-CMA experiment $\text{Exp}^{\text{euf-cma}}_{\mathsf{SPS},\mathcal{A}}(\lambda)$. Therefore, we have $\varepsilon_0 = \text{Adv}^{\text{euf-cma}}_{\mathsf{SPS},\mathcal{A}}(\lambda)$.

Transition $\mathbf{G}_0 \rightsquigarrow \mathbf{G}_0$: In game $\mathbf{G}_1$ we change the verification oracle to directly verify $[\mathbf{u}]_1$ in the group, instead of using the pairing product equation (for more details see Figure Fig. 4.11).

Note that a pair $(\mu^\star, \sigma^\star)$ that passes $\mathcal{O}_{\text{ver}}$ in $\mathbf{G}_1$ always passes the $\mathcal{O}_{\text{ver}}$ check in $\mathbf{G}_0$. Thus, to bound $|\varepsilon_0 - \varepsilon_1|$, it suffices to bound the probability that $\mathcal{A}$ produces a tuple $(\mu^\star, \sigma^\star)$ that passes $\mathcal{O}_{\text{ver}}$ in $\mathbf{G}_0$, but not in $\mathbf{G}_1$. For the

signature $\sigma^\star =: ([\mathbf{t}]_1, \Pi, [\mathbf{u}]_1)$ we can write the verification equation in $\mathbf{G}_0$ as

$$e([\mathbf{u}]_1^\top, [\mathbf{A}]_2) = e([\mathbf{t}]_1^\top, [\mathbf{K}_0\mathbf{A}]_2) + e(\begin{bmatrix}\mathbf{m}\\1\end{bmatrix}_1^\top, [\mathbf{K}\mathbf{A}]_2)$$
$$\Leftrightarrow e([\mathbf{u}]_1 - [\mathbf{t}]_1^\top\mathbf{K}_0 - \begin{bmatrix}\mathbf{m}\\1\end{bmatrix}_1^\top\mathbf{K}, [\mathbf{A}]_2) = \mathbf{0}$$

Observe that for any $(\mu^\star, ([\mathbf{t}]_1, \Pi, [\mathbf{u}]_1))$ that passes the verification equation in game $\mathbf{G}_0$, but not the one in $\mathbf{G}_1$, the value

$$[\mathbf{u}]_1 - [\mathbf{t}]_1^\top\mathbf{K}_0 - \begin{bmatrix}\mathbf{m}\\1\end{bmatrix}_1^\top\mathbf{K}$$

is a non-zero vector in the kernel of $\mathbf{A}$. Thus, from $\mathcal{A}$ we can construct an adversary $\mathcal{B}$ against the $\mathcal{D}_k$-KMDH assumption. Finally, Lemma 9 yields an adversary $\mathcal{B}'$ with $T(\mathcal{B}') \approx T(\mathcal{A}) + Q \cdot \mathsf{poly}(\lambda)$ such that

$$|\varepsilon_0 - \varepsilon_1| \leq \mathrm{Adv}^{\mathrm{mddh}}_{\mathcal{PG},\mathbb{G}_2,\mathcal{D}_k,\mathcal{B}}(\lambda).$$

Transition $\mathbf{G}_1 \rightsquigarrow \mathbf{G}_2$: We can replace $\mathbf{K}_0$ by $\mathbf{K}_0 + \mathbf{k}_0(\mathbf{a}^\perp)^\top$ for $\mathbf{a}^\perp \in \mathrm{orth}(\mathbf{A})$ and $\mathbf{k}_i \leftarrow_R \mathbb{Z}_p^{2k}$, as both are distributed identically. Note that this change does not show up in the public key pk. Looking ahead, this change will allow us to use the computational core lemma (Theorem 42). This yields

$$\varepsilon_1 = \varepsilon_2.$$

Transition $\mathbf{G}_2 \rightsquigarrow \mathbf{G}_3$: Let $\mathcal{A}$ be an adversary playing either $\mathbf{G}_2$ or $\mathbf{G}_3$. We build an adversary $\mathcal{B}$ such that $T(\mathcal{B}) \approx T(\mathcal{A}) + Q \cdot \mathsf{poly}(\lambda)$ and

$$\Pr[\mathrm{Exp}^{\mathrm{core}}_{0,\mathcal{B}}(1^\lambda) = 1] = \varepsilon_2 \quad \text{and} \quad \Pr[\mathrm{Exp}^{\mathrm{core}}_{1,\mathcal{B}}(1^\lambda) = 1] = \varepsilon_3.$$

This implies, by the core lemma (Theorem 42), that

$$\varepsilon_2 \leq \varepsilon_3 + \Delta^{\mathrm{core}}_{\mathcal{B}}(\lambda).$$

We now describe $\mathcal{B}$ against $\mathrm{Exp}^{\mathrm{core}}_{\beta,\mathcal{B}}(1^\lambda)$ for β equal to either 0 or 1. First, $\mathcal{B}$ receives $\mathsf{pp} := (\mathcal{PG}, [\mathbf{A}_0]_1, \mathsf{crs})$ from $\mathrm{Exp}^{\mathrm{core}}_{\beta,\mathcal{B}}(1^\lambda)$, then, $\mathcal{B}$ samples $\mathbf{A} \leftarrow_R \mathcal{D}_k$, $\mathbf{a}^\perp \in \mathrm{orth}(\mathbf{A})$, $\mathbf{K}_0 \leftarrow_R \mathbb{Z}_p^{2k\times(k+1)}$, $\mathbf{K} \leftarrow_R \mathbb{Z}_p^{(n+1)\times(k+1)}$ and forwards $\mathsf{pk} := (\mathcal{PG}, [\mathbf{A}_0]_1, \mathsf{crs}, [\mathbf{A}]_2, [\mathbf{K}_0\mathbf{A}]_2, [\mathbf{K}\mathbf{A}]_2)$ to $\mathcal{A}$.

To simulate $\mathcal{O}_{\mathrm{sign}}([\mathbf{m}]_1)$, $\mathcal{B}$ uses its oracle $\mathcal{O}_{\mathrm{tag}}$, which takes no input, and gives back $([\mathbf{t}]_1, \Pi, [u]_1)$. Then, $\mathcal{B}$ computes $[\mathbf{u}]_1 := \mathbf{K}_0^\top[\mathbf{t}]_1 + \mathbf{a}^\perp[u]_1 + \mathbf{K}^\top\begin{bmatrix}\mathbf{m}\\1\end{bmatrix}_1$, and returns $\sigma := ([\mathbf{t}]_1, \Pi, [\mathbf{u}]_1)$ to $\mathcal{A}$.

Finally, given the forgery $\left([\mathbf{m}^\star]_1, \sigma^\star\right)$ with signature $\sigma^\star := ([\mathbf{t}^\star]_1, \Pi^\star, [\mathbf{u}^\star]_1)$, $\mathcal{B}$ first checks if $[\mathbf{m}^\star]_1 \notin \mathcal{Q}_{\mathrm{sign}}$ and $[\mathbf{u}^\star]_1 \neq [\mathbf{0}]_1$. If it is not the case, then $\mathcal{B}$ returns 0 to $\mathcal{A}$. If it is the case, with the knowledge of $\mathbf{a}^\perp \in \mathbb{Z}_p$, $\mathcal{B}$ efficiently checks whether there exists $[u^\star]_1 \in \mathbb{G}_1$ such that $[\mathbf{u}^\star]_1 - \mathbf{K}_0^\top[\mathbf{t}^\star]_1 - \mathbf{K}^\top\begin{bmatrix}\mathbf{m}^\star\\1\end{bmatrix}_1 = [u^\star]_1\mathbf{a}^\perp$. If it is not the case, $\mathcal{B}$ returns 0 to $\mathcal{A}$. If it is the case, $\mathcal{B}$ computes $[u^\star]_1$ (it can do so efficiently given $\mathbf{a}^\perp$), sets $\mathsf{tag} := ([\mathbf{t}^\star]_1, \Pi^\star, [u^\star]_1)$, calls its verification oracle $\mathcal{O}_{\mathrm{ver}}(\mathsf{tag})$, and forwards the answer to $\mathcal{A}$.

Transition $\mathbf{G}_3 \rightsquigarrow \mathbf{G}_4$: In game $\mathbf{G}_3$ the vectors $\mathbf{r}$ sampled by $\mathcal{O}_{\text{sign}}$ are uniformly random over $\mathbb{Z}_p^k$, while they are uniformly random over $(\mathbb{Z}_p^k)^* = \mathbb{Z}_p^k \backslash \{0\}$ in $\mathbf{G}_4$. Since this is the only difference between the games, the difference of advantage is bounded by the statistical distance between the two distributions of $\mathbf{r}$. A union bound over the number of queries yields

$$\varepsilon_3 - \varepsilon_4 \leq \frac{Q}{p^k}.$$

Transition $\mathbf{G}_4 \rightsquigarrow \mathbf{G}_5$: These games are the same except for the extra condition $\widetilde{\mathsf{ctr}} = \mathsf{ctr}'$ in $\mathbf{G}_5$, which happens with probability $\frac{1}{Q}$ over the choice of $\widetilde{\mathsf{ctr}} \leftarrow_R [Q]$. Since the adversary view is independent of $\widetilde{\mathsf{ctr}}$, we have

$$\varepsilon_5 = \frac{\varepsilon_4}{Q}.$$

Game $\mathbf{G}_5$: We prove that $\varepsilon_5 \leq \frac{1}{p}$.

First, we can replace $\mathbf{K}$ by $\mathbf{K} + \mathbf{v}(\mathbf{a}^\perp)^\top$ for $\mathbf{v} \leftarrow_R \mathbb{Z}_p^{n+1}$, and $\{\mathbf{F}(i) : i \in [Q], i \neq \widetilde{\mathsf{ctr}}\}$ by $\{\mathbf{F}(i) + \mathbf{w}_i : i \in [Q], i \neq \widetilde{\mathsf{ctr}}\}$ for $\mathbf{w}_i \leftarrow_R \mathbb{Z}_p^{2k}$. Note that this does not change the distribution of the game.

Thus, for the i-th signing query with $i \neq \widetilde{\mathsf{ctr}}$ the value $\mathbf{u}$ is computed by $\mathcal{O}_{\text{sign}}([\mathbf{m}_i]_1)$ as

$$[\mathbf{u}]_1 = \mathbf{K}_0^\top [\mathbf{t}]_1 + (\mathbf{K}^\top + \mathbf{a}^\perp \mathbf{v}^\top) \begin{bmatrix} \mathbf{m}_i \\ 1 \end{bmatrix}_1 + \mathbf{a}^\perp (\mathbf{k}_0 + \mathbf{F}(i) + \mathbf{w}_i)^\top [\mathbf{t}]_1,$$

with $[\mathbf{t}]_1 := [\mathbf{A}_0]_1 \mathbf{r}$, $\mathbf{r} \leftarrow_R (\mathbb{Z}_p^k)^*$. This is identically distributed to

$$[\mathbf{u}]_1 = \mathbf{K}_0^\top [\mathbf{t}]_1 + \mathbf{K}^\top \begin{bmatrix} \mathbf{m}_i \\ 1 \end{bmatrix}_1 + \gamma_i \cdot [\mathbf{a}^\perp]_1, \text{ with } \gamma_i \leftarrow_R \mathbb{Z}_p.$$

For the $\widetilde{\mathsf{ctr}}$'th signing query, we have

$$[\mathbf{u}]_1 = \mathbf{K}_0^\top [\mathbf{t}]_1 + (\mathbf{K}^\top + \mathbf{a}^\perp \mathbf{v}^\top) \begin{bmatrix} \mathbf{m}_{\widetilde{\mathsf{ctr}}} \\ 1 \end{bmatrix}_1 + \mathbf{a}^\perp (\mathbf{k}_0 + \mathbf{F}(\widetilde{\mathsf{ctr}}))^\top [\mathbf{t}]_1.$$

Assuming $\mathcal{A}$ succeeds in producing a valid forgery, $\mathcal{O}_{\text{ver}}$ computes

$$[\mathbf{u}^\star]_1 = \mathbf{K}_0^\top [\mathbf{t}^\star]_1 + (\mathbf{K}^\top + \mathbf{a}^\perp \mathbf{v}^\top) \begin{bmatrix} \mathbf{m}^\star \\ 1 \end{bmatrix}_1 + \mathbf{a}^\perp (\mathbf{k}_0 + \mathbf{F}(\widetilde{\mathsf{ctr}}))^\top [\mathbf{t}]_1.$$

Since $\mathbf{m}^\star \neq \mathbf{m}_{\widetilde{\mathsf{ctr}}}$ by definition of the security game, we can use the pairwise independence of $\mathbf{m} \mapsto \mathbf{v}^\top \begin{bmatrix} \mathbf{m} \\ 1 \end{bmatrix}_1$ to argue that $\mathbf{v}^\top \begin{bmatrix} \mathbf{m}^\star \\ 1 \end{bmatrix}_1$ and $\mathbf{v}^\top \begin{bmatrix} \mathbf{m}_{\widetilde{\mathsf{ctr}}} \\ 1 \end{bmatrix}_1$ are two independent values, uniformly random over $\mathbb{G}_1$. Thus, the verification equation is satisfied with probability at most $\frac{1}{p}$, that is

$$\varepsilon_5 \leq \frac{1}{p}.$$

□

Bilateral Structure-Preserving Signature Scheme. Our structure-preserving signature scheme, SPS, defined in Figure 4.10 can sign only messages from $\mathbb{G}_1^n$. By applying the generic transformation from [KPW15, Section 6], we can transform our SPS to sign messages from $\mathbb{G}_1^{n_1} \times \mathbb{G}_2^{n_2}$ using their two-tier SPS, which is a generalization of [ACD+12]. The transformation is tightness-preserving by Theorem 6 of [KPW15] and costs additional k elements from $\mathbb{G}_1$ and $k+1$ elements from $\mathbb{G}_2$ in the signature. For the SXDH assumption ($k = 1$), our bilateral SPS scheme requires additional 1 element from $\mathbb{G}_1$ and 2 elements from $\mathbb{G}_2$ in the signature.

Chapter 5

Homomorphic Secret Sharing

From this chapter we focus on succinct secure communication. One approach towards secure communication with minimal communication complexity is via homomorphic secret sharing. Based on our work [BKS19], we present the first homomorphic secret sharing scheme from lattices that does not require expensive multiplications on ciphertexts, and show that this can lead to improvements regarding both communication complexity and computational efficiency in protocols for special-purpose secure 2-party computation and 2-server private information retrieval. We start this chapter by giving a technical overview of the techniques involved.

The central idea of our work is to leverage the special structure that most lattice-based encryption schemes have in common. More precisely, we observe that decryption typically consists of two steps: First, a function linear in the secret key is applied to the ciphertexts. Next, the result is recovered by rounding to the closest multiple of q/r, where q corresponds to the ciphertext modulus and r to the plaintext modulus. We refer to this property as *nearly linear decryption*.

Definition 47 (Informal - *Nearly Linear Decryption* [BKS19])**.** Let $R = \mathbb{Z}[X]/(X^N + 1)$ for N a power of 2. Let $r, q \in \mathbb{N}$ be moduli with $r|q$ and $1 \ll r \ll q$. We say that an encryption scheme supports *nearly linear decryption* for messages $m \in R_r := R/rR$ if the secret key is $\mathbf{s} \in R_q^d$, and for any ciphertext $\mathbf{c} \in R_q^d$ encrypting m,

$$\langle \mathbf{s}, \mathbf{c} \rangle = (q/r) \cdot m + e \mod q$$

for some "small" noise $e \in R$.

This captures various lattice-based encryption schemes [Reg05, ACPS09, LPR10, BV11, BPR12, LS15]. For the most efficient instantiation, we build on the encryption scheme by Lyubashevsky et al. [LPR10] based on learning with errors over rings. Note that our construction also generalizes to encryption schemes where the message is encrypted in "low-order symbols" (see Remark 56).

We show that every public-key encryption scheme with nearly linear decryption satisfies two properties we refer to as *"KDM oracle"* and *"distributed decryption"*. The first property builds on an observation of [BV11], who noted that nearly linear decryption already implies key-dependent message (KDM) security. We leverage their observation to implement a so-called KDM oracle, which allows parties to secret-share their inputs without knowledge of the secret key itself.

KDM oracle. Everyone in knowledge of the public key can generate encryptions of *key-dependent* messages (more precisely, multiples of the secret key) *without* knowledge of the secret key itself.

For the second property, we build on two techniques we call *rounding* and *lifting*:

Rounding. If $r << q$, then local rounding[1] on 2-party random additive secret shares of $\approx (q/r) \cdot x \mod q$ for some $x \in R_r$ yields additive secret shares of $x \mod r$ with overwhelming probability [DHRW16].

Lifting. If $\|z\|_\infty << r$, then 2-party random additive secret shares of $z \mod r$ are random additive secret shares of $z \mod q$ (for arbitrary q) with overwhelming probability.

Note that while both observations may look simple at first glance, they are highly non-trivial: Both, rounding and lifting yield an error with *constant* probability in general (and in particular in the case of 3 or more parties).

Rounding allows to perform *one multiplication* via distributed decryption, resulting in shares of $x \cdot y \cdot \mathsf{sk} \mod r$. *Lifting* allows to "lift" the resulting shares back to shares $\mod q$, which is necessary to perform another multiplication (as ciphertexts "live" in R_q). Together, this yields the second property described in the following.

Multiplication via distributed decryption. There exists a procedure DDec, such that, roughly,

$$\mathsf{DDec}_{y\cdot\mathsf{sk}_1}(\mathbf{c}^{x\cdot\mathsf{sk}}) + \mathsf{DDec}_{y\cdot\mathsf{sk}_2}(\mathbf{c}^{x\cdot\mathsf{sk}}) = x \cdot y \cdot \mathsf{sk} \mod q,$$

where $\mathbf{c}^{x\cdot\mathsf{sk}}$ is an encryption of $x \cdot \mathsf{sk} \mod p$ and $\mathsf{sk}_1 + \mathsf{sk}_2 = \mathsf{sk} \mod q$ is an additive secret sharing of the secret key.

The *KDM-oracle* allows parties to *share* their inputs and *distributed decryption* allows parties to locally perform multiplications on shared values. Note that our techniques limit us to multiplications of an *encryption* (i.e. input value) with a shared *multiple of the secret key.* The class of programs captured are so-called *restricted straight-line multiplication (RMS)* programs, where memory values (in our case represented by shared multiples of the secret key) can only be multiplied by input values. Note that RMS programs can emulate branching programs and all circuits in NC^1. Our main result is summarized in the following theorem.

Theorem 48 (Informal - *Main HSS Construction* [BKS19])**.** *Given any encryption scheme with nearly linear decryption (as above) over ring R with parameters $r, q, d \in \mathbb{N}$, as well as magnitude bound $B \in \mathbb{N}$ for which $B \ll r \ll q/B$, and output modulus $\beta \leq B$, there exists 2-party public-key HSS for inputs in R_β with size:*

- *Public key $\mathsf{pk} = pk$ of the encryption scheme, Evaluation keys $\mathsf{ek}_b \in R_q^{d-1}$*
- *HSS shares of each input $x_i \in R_\beta$ consist of d ciphertexts.*

supporting any polynomial number of the following homomorphic operations over R_β (subject to the ℓ_∞ magnitude bound $\|y\|_\infty \leq B$ (in R) for all partial computation values y), with negligible correctness error, and the specified complexities:

- *Loading an input into memory ($y_i \leftarrow x_i$):* *d decryptions*
- *Addition of memory values ($y_k \leftarrow y_i + y_j$):* *1 addition over R_q^d*

[1] Here, by rounding we denote returning the closest multiple of (q/r).

- *Multiplication of input with memory value* ($y_k \leftarrow x_i \cdot y_j$): *d decryptions*
  "Terminal" multiplication (s.t. y_k *appears in no future mult):* *1 decryption*

where "decryption" is essentially one inner product over R_q^d.

Note that for instantiations from learning with errors over rings we have $d = 2$.

Roadmap. We begin this chapter by giving a formal definition of RMS programs in Section 5.1. Next, we introduce decryption with nearly linear decryption and derive the required properties in Sections 5.2 and 5.3. We present our homomorphic secret sharing scheme in Section 5 and give a brief overview of extensions in Section 5.5. In Section 5.6, we show how to instantiate our scheme and give efficiency estimates. Finally, in Section 5.7 we show the practical applicability of our scheme within the applications of secure 2-party computation of low-degree polynomials, and 2-server generalized private information retrieval.

The following is taken in large parts verbatim from our work [BKS19].

5.1 Computational Models

Our main HSS scheme naturally applies to programs P in a computational model known as *Restricted Multiplication Straight-line (RMS)* programs [Cle91, BGI16a].

Definition 49 (RMS programs)**.** An RMS program consists of a magnitude bound $B_{\max}$ and an arbitrary sequence of the four following instructions, sorted according to a unique identifier $\mathrm{id} \in \mathcal{S}_{\mathrm{id}}$:

- Load an input into memory: $(\mathrm{id}, \widehat{y}_j \leftarrow \widehat{x}_i)$.
- Add values in memory: $(\mathrm{id}, \widehat{y}_k \leftarrow \widehat{y}_i + \widehat{y}_j)$.
- Add input values: $(\mathrm{id}, \widehat{x}_k \leftarrow \widehat{x}_i + \widehat{x}_j)$.
- Multiply memory value by input: $(\mathrm{id}, \widehat{y}_k \leftarrow \widehat{x}_i \cdot \widehat{y}_j)$.
- Output from memory, as R element: $(\mathrm{id}, \beta, \widehat{O}_j \leftarrow \widehat{y}_i)$.

If at any step of execution the size of a memory value exceeds the bound $B_{\max}$ (i.e. $\|\widehat{y}_j\|_\infty > B_{\max}$), the output of the program on the corresponding input is defined to be $\perp$. Otherwise the output is the sequence of $\widehat{O}_j$ values, sorted by id. We define the *size* (resp., *multiplicative size*) of an RMS program P as the number of instructions (resp., multiplication and load input instructions). Note that we consider addition of input values merely for the purpose of efficiency. We denote the maximum number of additions on input values by $P_{\mathrm{inp+}}$.

5.2 Encryption with Nearly Linear Decryption

We now formally introduce encryption with nearly linear decryption. Basically, we require the following properties: First, there is a way to encrypt certain key-dependent messages without knowledge of the secret key. Second, it is possible to "distributively decrypt" a ciphertext. More precisely, given an encryption of message x and secret shares of some multiple y of the secret key $\mathsf{sk} = \mathbf{s}$, there is a way to

obtain secret shares of $x \cdot y$ *over the same modulus as the original secret shares.* This enables us to perform several stages of distributed decryption. That is, given an encryption of $x \cdot \mathbf{s}$ (for some value x) and a secret share of $y \cdot \mathbf{s}$ modulo q, distributed decryption results in a secret share of $x \cdot y \cdot \mathbf{s}$ modulo q, which can serve as input to another distributed decryption.

In the following we show that all public-key encryption scheme that satisfy *nearly linear decryption*, both support a KDM-oracle and homomorphic multiplication via distributed decryption.

Definition 50 (Encryption scheme with nearly linear decryption)**.** Let $\mathsf{PKE} := (\mathsf{PKE.Gen}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ be a public-key encryption scheme with pseudorandom ciphertexts. We say that PKE is a public-key *encryption scheme with nearly linear decryption* if it further satisfies the following properties:

- **Parameters:** The scheme is parametrized by modulus values $r, q \in \mathbb{N}$, dimension $d \in \mathbb{N}$, and bounds $B_{\mathrm{sk}}, B_{\mathrm{ct}} \in \mathbb{N}$, where $r|q$, $r \geq \lambda^{\omega(1)}$, $q/r \geq \lambda^{\omega(1)}$ and $d, B_{\mathrm{sk}}, B_{\mathrm{ct}} \leq \mathsf{poly}(\lambda)$, as well as a ring $R = \mathbb{Z}[X]/(X^N+1)$, where $N \leq \mathsf{poly}(\lambda)$ is a power of 2.[2]
- **Message space and secret key:** The scheme has message space $\mathcal{M} := R_r := R/pR$ and ciphertext space $\mathcal{C} := R_q^d := (R/qR)^d$. The secret key $\mathbf{s}$ returned by $\mathsf{PKE.Gen}$ on input 1^λ is an element of R^d satisfying $\|\mathbf{s}\|_\infty \leq B_{\mathrm{sk}}$. Further, $\mathbf{s}$ is of the form $(1, \widehat{\mathbf{s}})$ for some $\widehat{\mathbf{s}} \in R_r^{d-1}$.
- **Nearly linear decryption:** For any $\lambda \in \mathbb{N}$, for any $(\mathsf{pk}, \mathbf{s})$ in the image of $\mathsf{Gen}(1^\lambda)$, for any message $x \in R_r$ and for any ciphertext $\mathbf{c} \in R_q^d$ in the image of $\mathsf{PKE.Enc}(\mathsf{pk}, x)$, for some $e \in R$ with $\|e\|_\infty \leq B_{\mathrm{ct}}$ it holds

$$\langle \mathbf{s}, \mathbf{c} \rangle = (q/r) \cdot x + e \mod q.$$

Remark 51. Encryption with nearly linear decryption can be instantiated based on LWE (e.g. with [Reg05, ACPS09], where $d = \lambda$) and based on RLWE (e.g. with [LPR10, BV11], where $d = 2$). Further, it can be instantiated with schemes based on assumptions like module-LWE [LS15] and LWR [BPR12]. For more details on the instantiation from the RLWE-based encryption scheme of LPR [LPR10], we refer to Section 5.6, and for the instantiation from the LWE-based encryption scheme of Regev [Reg05] we refer to the full version of [BKS19].

5.3 Properties of Encryption Schemes with Nearly Linear Decryption

We now prove that our two desired properties are satisfied by any encryption scheme with nearly linear decryption.

KDM-oracle. Recall that the first property allows anyone to compute an encryption of any linear function of the secret key without having access to the secret key itself, serving as a "KDM oracle." A similar notion, but for secret-key encryption schemes and with deterministic procedure, was introduced in [BDH14].

[2] To simplify the analysis, we restrict the definition to 2-power cyclotomic rings. However, our construction can be generalized to arbitrary cyclotomics.

$\underline{\mathrm{Exp}^{\mathrm{kdm-ind}}_{\mathsf{PKE.OKDM},\mathcal{A}}(\lambda):}$	$\underline{\mathcal{O}_{\mathrm{kdm}}(x,j):}$
$(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{PKE.Gen}(1^\lambda)$	**if** $\beta = 0$
$\beta \leftarrow \{0,1\}$	**return** $\mathsf{PKE.OKDM}(\mathsf{pk},x,j)$
$\beta' \leftarrow \mathcal{A}^{\mathcal{O}_{\mathrm{kdm}}(\cdot,\cdot)}(1^\lambda,\mathsf{pk})$	**else**
if $\beta = \beta'$ **return** 1	**return** $\mathsf{PKE.Enc}(\mathsf{pk},0)$
else return 0	

Figure 5.1: Security challenge experiment for the KDM oracle.

Lemma 52 (KDM oracle)**.** *Let* $\mathsf{PKE} := (\mathsf{PKE.Gen},\mathsf{PKE.Enc},\mathsf{PKE.Dec})$ *be a public-key encryption scheme with nearly linear decryption and parameters* $(r,q,d,B_{\mathrm{sk}},B_{\mathrm{ct}},R)$.

Then, for the PPT procedure $\mathsf{PKE.OKDM}$ *that on input of a public key* pk*, a value* $x \in R$ *and an index* $j \in \{1,\ldots,d\}$ *computes an encryption* $\mathbf{c} \leftarrow \mathsf{PKE.Enc}(\mathsf{pk},0)$ *and outputs*

$$\mathbf{c}_j := (q/r)\cdot x \cdot \mathbf{e}_j + \mathbf{c} \mod q,$$

where $\mathbf{e}_j \in R_q^d$ *is the* j*-th unit vector, the following properties are satisfied.*

- ***Nearly linear decryption to the message*** $x \cdot s_j$***:*** *For any* $\lambda \in \mathbb{N}$*, for any* $(\mathsf{pk},\mathbf{s})$ *in the image of* $\mathsf{Gen}(1^\lambda)$*, and for any ciphertext* $\mathbf{c}_j \in R_q^d$ *in the image of* $\mathsf{PKE.OKDM}(\mathsf{pk},x,j)$*, it holds*

  $$\langle \mathbf{s},\mathbf{c}_j\rangle = (q/r)\cdot(x\cdot s_j) + e \mod q$$

  for some $e \in R$ *with* $\|e\|_\infty \le B_{\mathrm{ct}}$.

- ***Security:*** *For any* $\lambda \in \mathbb{N}$ *and any PPT adversary* $\mathcal{A}$ *we have that*

  $$\mathrm{Adv}^{\mathrm{kdm-ind}}_{\mathsf{PKE.OKDM},\mathcal{A}}(\lambda) := \left|\Pr\left[\mathrm{Exp}^{\mathrm{kdm-ind}}_{\mathsf{PKE.OKDM},\mathcal{A}}(\lambda)=1\right] - 1/2\right|$$

  is negligible in λ*, where* $\mathrm{Exp}^{\mathrm{kdm-ind}}_{\mathsf{PKE.OKDM},\mathcal{A}}(\lambda)$ *is as defined in Figure 5.1*

Proof. We have to prove that $\mathsf{PKE.OKDM}$ meets the required properties. As PKE is a encryption scheme with nearly linear decryption, we have

$$\mathbf{c} = (q/r)\cdot 0 + e = e \mod q$$

for some $e \in R$ with $\|e\|_\infty \le B_{\mathrm{ct}}$. We thus have

$$\langle \mathbf{s},\mathbf{c}_j\rangle = (q/r)\cdot x\cdot\langle \mathbf{s},\mathbf{e}_j\rangle + \langle \mathbf{s},\mathbf{c}\rangle = (q/r)\cdot(x\cdot s_j)+e \mod q,$$

as required for nearly linear decryption.

In order to prove security of $\mathsf{PKE.OKDM}$, we proceed via a series of games depicted in Figure 5.2. We have

$$\mathrm{Adv}^{\mathrm{kdm-ind}}_{\mathsf{PKE.OKDM},\mathcal{A}}(\lambda) := \left|\Pr\left[\mathbf{G}_0=1\right] - \frac{1}{2}\right|.$$

From an adversary distinguishing between $\mathbf{G}_0$ and $\mathbf{G}_1$, we can construct an adversary $\mathcal{B}$ on the pseudorandomness of ciphertexts with

$$|\Pr[\mathbf{G}_0=1] - \Pr[\mathbf{G}_1=1]| \le \mathrm{Adv}^{\mathrm{pr}}_{\mathsf{PKE},\mathcal{B}}(\lambda).$$

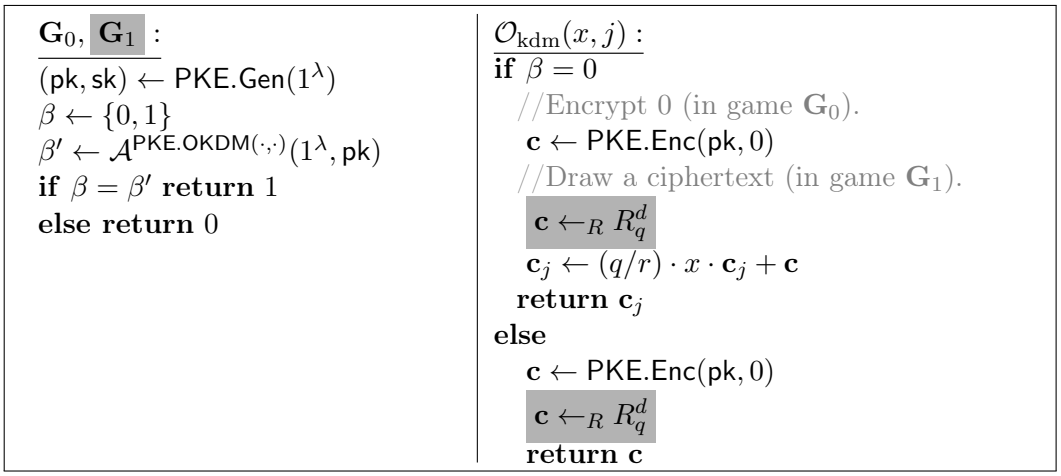

Figure 5.2: Games $\mathbf{G}_0$ and $\mathbf{G}_1$ in the proof of Lemma 52 (Security of OKDM).

In game $\mathbf{G}_1$ the view of $\mathcal{A}$ is independent of β, as the vectors $\mathbf{c}_j$ and $\mathbf{c}$ are both distributed uniformly at random over R_r^d. We have thus

$$\Pr[\mathbf{G}_1 = 1] = \frac{1}{2}.$$

As PKE satisfies indistinguishability of ciphertexts by prerequisites, we have thus that

$$\mathrm{Adv}^{\mathrm{kdm-ind}}_{\mathsf{PKE.OKDM},\mathcal{A}}(\lambda) \leq \mathrm{Adv}^{\mathrm{pr}}_{\mathsf{PKE},\mathcal{B}}(\lambda)$$

is negligible in λ as required. □

Rounding and lifting. First, we present a local rounding trick as in [DHRW16] which allows to recover the shares of x (modulo r). The idea is that if q/r is large, the probability that the error term e leads to a rounding error is small. Note that it is crucial here that we are in the 2-party setting, where the secret shares of $(q/r) \cdot x + e \mod q$ have both approximately (that is, except for the error e) the same distance from some multiple of q/r. In fact, even for arbitrarily large gap between r and q, rounding for 3 or more parties fails with constant probability.

Lemma 53 (Rounding of noisy shares)**.** *Let $r, q \in \mathbb{N}$ be modulus values with $q/r \geq \lambda^{\omega(1)}$. Let $R = \mathbb{Z}[X]/(X^N + 1)$ for N a power of 2 (i.e. $N = 2^n$ for $n \in \mathbb{N}_0$). Let $t_0, t_1 \in R_q$ random subject to*

$$t_0 + t_1 = (q/r) \cdot x + e \mod q$$

for some $x \in R_r, e \in R$ with $q/(r \cdot \|e\|_\infty) \geq \lambda^{\omega(1)}$. Then, for the the deterministic polynomial time procedure Round *that on input $t_b \in R_q$ outputs*

$$\lfloor (r/q) \cdot t_b \rceil \mod r \in R_r$$

it holds:

$$\mathsf{Round}(t_0) + \mathsf{Round}(t_1) = x \mod r$$

with probability at least $1 - N \cdot (\|e\|_\infty + 1) \cdot r/q \geq 1 - \lambda^{-\omega(1)}$ over the choice of the shares t_0, t_1.

Proof. In order to prove correctness of Round we consider the values of t_0 and t_1 as elements in R. We express t_0 in the basis of (q/r), i.e. let $I := [-q/(2p), q/(2p))$, let $z_0 \in R_r$, $r_0 \in R|_I$ such that $t_0 = (q/r) \cdot z_0 + r_0$. Let $l \in R$ such that $t_1 = (q/r) \cdot (x - z_0) + e - r_0 + q \cdot l$ in R. Then we have

$$\begin{aligned}\mathsf{Round}(t_0) &= \lfloor (r/q) \cdot ((q/r) \cdot z_0 + r_0) \rceil \mod r \\ &= \lfloor z_0 + \underbrace{(r/q) \cdot r_0}_{\in R|_{[-1/2,1/2)}} \rceil \mod r \\ &= z_0 \mod r\end{aligned}$$

and

$$\begin{aligned}\mathsf{Round}(t_1) &= \lfloor (r/q) \cdot ((q/r) \cdot (x - z_0) + e - r_0 + q \cdot l) \rceil \mod r \\ &= \lfloor x - z_0 + r \cdot l + (r/q) \cdot (e - r_0) \rceil \mod r.\end{aligned}$$

Now, assume $e - r_0 \in R|_{[-q/(2p),q/(2p))}$. In this case it holds

$$\begin{aligned}\mathsf{Round}(t_1) &= \lfloor x - z_0 + r \cdot l + \underbrace{(r/q) \cdot (e - r_0)}_{\in R|_{[-1/2,1/2)}} \rceil \mod r \\ &= x - z_0 \mod r.\end{aligned}$$

It is left to compute the probability of $e - r_0 \in R|_I$. This is the case, whenever all coefficients of r_0 are not "too close" to the boundaries of the interval I (constituting the *good area*). As t_0 is chosen uniformly at random, we have that the distribution of z_0 is the uniform distribution over R_r and the distribution of r_0 is the uniform distribution over $R|_I$. For every $j \in \{1, \ldots, N\}$, for the j-th component of r_0 the probability that it is outside the interval

$$I_j := (-q/(2p) + e_j, q/(2p) + e_j]$$

is at most

$$(\|e\|_\infty + 1) \cdot r/q.$$

A union bound over all components of r_0 yields thus correct rounding with probability at least

$$1 - N \cdot (\|e\|_\infty + 1) \cdot r/q.$$

□

The following simple observation constitutes a crucial step of our HSS construction, as it will allow to have several levels of multiplication without requiring a sequence of decreasing moduli. While in general the conversion of secret shares from one modulus to another constitutes a problem, we observe that whenever the secret shared value is *small* in comparison to the modulus, and we use the centered representation of R_r with coefficients in $(-\lfloor r/2 \rceil, \ldots, \lfloor (r-1)/2 \rceil]$, then with high probability the secret sharing actually constitutes a secret sharing over R, so switching to an arbitrary modulus is trivial. Note that (as for rounding) this only holds true in the 2-party setting.

Lemma 54 (Lifting the modulus of shares)**.** *Let $r \in \mathbb{N}$ be a modulus with $r \geq \lambda^{\omega(1)}$. Let $R = \mathbb{Z}[X]/(X^N + 1)$ for N a power of 2. Let $x \in R$ and $z_0, z_1 \in R_r$ be random, subject to*

$$z_0 + z_1 = x \mod r.$$

Then, we have

$$z_0 + z_1 = x \text{ over } R$$

with probability at least $1 - (N \cdot (\|x\|_\infty + 1)/r) \geq 1 - \lambda^{-\omega(1)}$ over the choice of the shares z_0, z_1.

Proof. We have to show that for $z_0 \leftarrow_R R_r$ random, with overwhelming probability it holds $x - z_0 \in R_r$ (without computing modulo r). Recall that we consider R_r as elements whose coefficients are all in $I := (-\lfloor r/2 \rceil, \ldots, \lfloor (r-1)/2 \rceil]$. Thus, $x - z_0 \in R_r$, whenever for all $j = \{1, \ldots, N\}$, the j-th coefficient of z_0 is in $I_j := [-\lfloor (r-1)/2 \rceil + x_j, \ldots, \lfloor r/2 \rceil + x_j)$. For every j we have $|I \cap I_j| \geq r - x_j - 1$. A union bound over all coefficients yields that $x - z_0 \in R_r$ (and thus $z_0 + z_1 = x$ over R) except with probability at most

$$N \cdot (\|x\|_\infty + 1)/r.$$

□

Multiplication via distributed decryption. The following shows that any encryption with nearly linear decryption allows two parties to perform decryption *distributively*, employing their respective shares of the secret key to obtain a secret share of the corresponding message *modulo q*. Further, the scheme inherently supports homomorphic addition of ciphertexts, and the distributed decryption property holds accordingly for any sum of a bounded number of ciphertexts (generated from Enc or OKDM).

Lemma 55 (Distributed decryption of sums of ciphertexts)**.** *Let* $\mathsf{PKE} := (\mathsf{PKE.Gen}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ *be a public-key encryption scheme with nearly linear decryption and parameters $(r, q, d, B_{\mathrm{sk}}, B_{\mathrm{ct}}, R)$, where R has dimension N. Let* $\mathsf{PKE.OKDM}$ *be the KDM oracle from Lemma 52. Let $B_{\mathrm{add}} \in \mathbb{N}$ with $B_{\mathrm{add}} \leq \mathsf{poly}(\lambda)$. Then the deterministic polynomial time decryption procedure* $\mathsf{PKE.DDec}$ *that on input $b \in \{0,1\}, \mathbf{t}_b \in R^d, \mathbf{c} \in R_q^d$ outputs*

$$\mathsf{Round}(\langle \mathbf{t}_b, \mathbf{c} \rangle \mod q) \in R_q$$

(where Round *is as in Lemma 53) satisfies the following:*

For all $y \in R_r$ with $r/\|y\|_\infty \geq \lambda^{\omega(1)}$ and $q/(r \cdot \|y\|_\infty) \geq \lambda^{\omega(1)}$, for all $(\mathsf{pk}, \mathbf{s}) \leftarrow \mathsf{Gen}(1^\lambda)$, for all messages $x_1 \ldots, x_{B_{\mathrm{add}}} \in R_r$ with $r/\|x_i\|_\infty \geq \lambda^{\omega(1)}$, for all encryptions $\mathbf{c}_i$ of x_i that are either output of $\mathsf{PKE.Enc}$ *or of* $\mathsf{PKE.OKDM}$ *and for shares $\mathbf{t}_0, \mathbf{t}_1 \in R_q^d$ random subject to*

$$\mathbf{t}_0 + \mathbf{t}_1 = y \cdot \mathbf{s} \mod q$$

for $\mathbf{c} := \sum_{i=1}^{B_{\mathrm{add}}} \mathbf{c}_i$ and $x := \sum_{i=1}^{B_{\mathrm{add}}} x_i$ it holds

$$\mathsf{PKE.DDec}(0, \mathbf{t}_0, \mathbf{c}) + \mathsf{PKE.DDec}(1, \mathbf{t}_1, \mathbf{c}) = x \cdot y \mod q$$

with probability over the random choice of the shares $\mathbf{t}_0, \mathbf{t}_1$ of at least

$$1 - N \cdot (N \cdot B_{\mathrm{add}} \cdot \|y\|_\infty \cdot B_{\mathrm{ct}} \cdot r/q + \|x \cdot y\|_\infty / r + r/q + 1/r) \geq 1 - \lambda^{-\omega(1)}.$$

Proof. The idea of the proof is that nearly linear decryption allows (almost) homomorphic addition of ciphertexts with linear growth in the error. As $q/(r \cdot \|y\|_\infty) \geq \lambda^{\omega(1)}$ and the vectors $\mathbf{t}_b$ are individually random, by Lemma 53 we can recover $x \cdot y \mod r$ with overwhelming probability. Finally, as $r \geq \lambda^{\omega(1)}$, by Lemma 54 we can *lift* the modulus q (as with overwhelming probability the shares constitute a correct sharing of $x \cdot y$ over R and thus R_q).

We start by proving that nearly linear decryption holds true also for the sum of a bounded number of ciphertexts, but with an increased error term. To that end, let $x := \sum_{i=1}^{B_{\text{add}}} x_i$. Because PKE is an encryption scheme with nearly linear decryption and by Lemma 52, for all $i \in \{1, \ldots, B_{\text{add}}\}$ we have

$$\langle \mathbf{s}, \mathbf{c}_i \rangle = (q/r) \cdot x_i + e_i \mod q, \tag{5.1}$$

where $\|e_i\|_\infty \leq B_{\text{ct}}$. This yields

$$\langle \mathbf{s}, \sum_{i=1}^{B_{\text{add}}} \mathbf{c}_i \rangle = \sum_{i=1}^{B_{\text{add}}} \langle \mathbf{s}, \mathbf{c}_i \rangle = \sum_{i=1}^{B_{\text{add}}} ((q/r) \cdot x_i + e_i) = (q/r) \cdot x + \sum_{i=1}^{B_{\text{add}}} e_i \mod q,$$

where $\| \sum_{i=1}^{B_{\text{add}}} e_i \|_\infty \leq \sum_{i=1}^{B_{\text{add}}} \|e_i\|_\infty \leq B_{\text{add}} \cdot B_{\text{ct}}$.

It thus holds

$$\langle \mathbf{t}_0, \mathbf{c} \rangle + \langle \mathbf{t}_1, \mathbf{c} \rangle = y \cdot \langle \mathbf{s}, \mathbf{c} \rangle = (q/r) \cdot (x \cdot y) + (e \cdot y) \mod q$$

for some $e \in R$ with $\|e\|_\infty \leq B_{\text{add}} \cdot B_{\text{ct}}$.

By Lemma 53 we have

$$\mathsf{Round}(\langle \mathbf{t}_0, \mathbf{c} \rangle \mod q) + \mathsf{Round}(\langle \mathbf{t}_1, \mathbf{c} \rangle \mod q) = y \cdot x \mod r$$

except with probability at most

$$N \cdot (\|y \cdot e\|_\infty + 1) \cdot r/q \quad \leq \quad N \cdot (N \cdot B_{\text{add}} \cdot \|y\|_\infty \cdot B_{\text{ct}} + 1) \cdot r/q.$$

Whenever Round is successful, by Lemma 54 we have

$$(\mathsf{Round}(\langle \mathbf{t}_0, \mathbf{c} \rangle \mod q)) + (\mathsf{Round}(\langle \mathbf{t}_1, \mathbf{c} \rangle \mod q)) = x \cdot y \mod q$$

except with probability at most $N \cdot (\|x \cdot y\|_\infty + 1)/r$. □

Remark 56. Note that our techniques also extend to encryption schemes which encrypt messages in low-order symbols, e.g. where $\langle \mathbf{s}, \mathbf{c} \rangle = x + r \cdot e \mod q$ for r and q coprime. As mentioned in the main part, our techniques carry over to encryption schemes

For this class of encryption schemes, we define $\mathsf{PKE.OKDM}$ as the algorithm that on input of a public key pk, a value $y \in R$ with $\|y\|_\infty \leq B_{\text{inp}}$ and an index $j \in \{1, \ldots, d\}$ computes an encryption $\mathbf{c} \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}, 0)$ and outputs $\mathbf{c}_j := y \cdot \mathbf{e}_j + \mathbf{c} \mod q$, where $\mathbf{e}_j \in R_q^d$ is the j-th unit vector.

Further, we define $\mathsf{PKE.DDec}$ to be the algorithm that on input $b \in \{0, 1\}, \mathbf{t}_b \in R^d, \mathbf{c} \in R_q^d$ outputs

$$(\langle \mathbf{t}_b, \mathbf{c} \rangle \mod q) \mod r) \mod q.$$

5.4 Our HSS from Encryption with Nearly Linear Decryption

We now present our construction of a public-key HSS from an encryption scheme with nearly linear decryption. For various extensions that allow to improve the efficiency in specific applications we refer to the full version of [BKS19].

Theorem 57 (HSS from encryption with nearly linear decryption)**.** *Let* $\mathsf{PKE} := (\mathsf{PKE.Gen}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ *be a secure public-key encryption scheme with nearly linear decryption and parameters* $(r, q, d, B_{\mathrm{sk}}, B_{\mathrm{ct}}, R)$*.*

- *Let* $B_{\mathrm{inp}} \in \mathbb{N}$ *with* $r/B_{\mathrm{inp}} \geq \lambda^{\omega(1)}$ *and* $q/(B_{\mathrm{inp}} \cdot r) \geq \lambda^{\omega(1)}$*.*
- *Let* $\mathsf{PKE.OKDM}$ *be the KDM oracle from Lemma 52.*
- *Let* $\mathsf{PKE.DDec}$ *be the distributed decryption from Lemma 55.*
- *Let* $\mathsf{PRF}\colon K \times \mathcal{S}_{\mathrm{id}} \to R_q^d$ *be a pseudorandom function.*

Then, the scheme $\mathsf{HSS} = (\mathsf{HSS.Gen}, \mathsf{HSS.Enc}, \mathsf{HSS.Eval})$ *given in Figure 5.3 is a* 2*-party public-key homomorphic secret sharing scheme with input space* $[R]_{B_{\mathrm{inp}}}$ *for the class of RMS programs with magnitude bound* $B_{\max}$*, where* $r/B_{\max} \geq \lambda^{\omega(1)}$ *and* $q/(B_{\max} \cdot r) \geq \lambda^{\omega(1)}$*. More precisely,* HSS *satisfies the following.*

- ***Correctness:*** *For any* $\lambda \in \mathbb{N}$*, for any* $x^{(1)}, \ldots, x^{(\rho)} \in [R]_{B_{\mathrm{inp}}}$*, for any polynomial-sized RMS program* P *with* $P(x^{(1)}, \ldots, x^{(\rho)}) \neq \perp$ *and magnitude bound* $B_{\max}$ *with* $r/B_{\max} \geq \lambda^{\omega(1)}$ *and* $q/(B_{\max} \cdot r) \geq \lambda^{\omega(1)}$*, and for any integer* $\beta \geq 2$*, there exist a PPT adversary* $\mathcal{B}$ *on the pseudorandomness of* PRF *such that*

$$\Pr^{\mathrm{cor}}_{\mathsf{HSS},(x^{(i)})_i,P,\beta}(\lambda) \geq 1 - \left(\mathrm{Adv}^{\mathrm{prf}}_{\mathsf{PRF},\mathcal{B}}(\lambda) + \lambda^{-\omega(1)}\right).$$

- ***Security:*** *For every PPT adversary* $\mathcal{A}$ *on the security of* HSS*, there exists an PPT adversary* $\mathcal{B}$ *on the security of* $\mathsf{PKE.OKDM}$ *such that*

$$\mathrm{Adv}^{\mathrm{sec}}_{\mathsf{HSS},\mathcal{A}}(\lambda) \leq \mathrm{Adv}^{\mathrm{kdm-ind}}_{\mathsf{PKE.OKDM},\mathcal{B}}(\lambda).$$

Proof. For the proof of correctness we refer to Lemma 58.

For the proof of security we employ a hybrid argument. We define the corresponding games in Figure 5.4. Game $\mathbf{G}_0$ corresponds to the HSS security game, therefore we have

$$\mathrm{Adv}^{\mathrm{sec}}_{\mathcal{A},\mathsf{HSS}}(\lambda) = \left|\Pr\left[\mathbf{G}_0 = 1\right] - 1/2\right|.$$

From a PPT adversary $\mathcal{A}$ distinguishing between $\mathbf{G}_0$ and $\mathbf{G}_1$ we can construct a PPT adversary $\mathcal{B}$ on the security of $\mathsf{PKE.OKDM}$ as follows. On input $(b, x_0, x_1, \mathsf{state})$ by $\mathcal{A}$ and input of the public key pk by the security challenge experiment of the KDM oracle, $\mathcal{B}$ chooses $\beta \in \{0,1\}$, $\mathbf{s}_0 \leftarrow_R R_q^d$, sets $\mathbf{s}_1 := \mathbf{s} - \mathbf{s}_0 \mod q$ and queries $\mathbf{c}_j \leftarrow \mathcal{O}_{\mathrm{kdm}}(x_\beta, j)$ for all $j \in \{1, \ldots, d\}$. Finally, $\mathcal{B}$ sends pk, $\mathsf{ek}_b := (K, \mathbf{s}_b)$ and $\mathbf{E}$ to $\mathcal{A}$, where $\mathbf{E} := (\mathbf{c}_1 | \ldots | \mathbf{c}_d) \in R_q^{d \times d}$. If the security challenge experiment of the KDM oracle returns real encryptions of $x \cdot s_j$, the distribution of ek_b equals the

$\underline{\mathsf{HSS.Gen}(1^\lambda):}$

- Generate a key pair $(\mathsf{pk}, \mathbf{s}) \leftarrow \mathsf{PKE.Gen}(1^\lambda)$ for encryption and draw a PRF key $K \leftarrow_R K$. //Recall $\mathbf{s} = (1, \widehat{\mathbf{s}}) \in R_q \times R_q^{d-1}$.
- **Secret share the secret key.** Choose $\mathbf{s}_0 \leftarrow_R R_q^d$ at random. Define

$$\mathbf{s}_1 := \mathbf{s} - \mathbf{s}_0 \mod q.$$

- Output pk and $\mathsf{ek}_b \leftarrow (K, \mathbf{s}_b)$.

$\underline{\mathsf{HSS.Enc}(1^\lambda, \mathsf{pk}, x):}$

- **Encrypt the input.** Compute and output $\mathbf{E}^x \leftarrow \mathsf{PKE.OKDM}(\mathsf{pk}, x)$. //This corresponds to $\mathbf{E}^x = \mathsf{PKE.Enc}(\mathsf{pk}, x \cdot \mathbf{s}) \in R_q^{d \times d}$.

$\underline{\mathsf{HSS.Eval}(b, \mathsf{ek}_b, (\mathbf{E}^{x^{(1)}}, \ldots, \mathbf{E}^{x^{(\rho)}}), P, \beta):}$

Parse $(K, \mathbf{s}_b) =: \mathsf{ek}_b$, parse P as a sequence of RMS operations and proceed as follows.

- **Load an input into memory:** On instruction $(\mathrm{id}, \mathbf{E}^x)$ compute

$$\mathbf{t}_b^x := \mathsf{PKE.DDec}(b, \mathbf{s}_b, \mathbf{E}^x) + (1 - 2b) \cdot \mathsf{PRF}(K, \mathrm{id}) \mod q.$$

- **Add values in memory:** On instruction $(\mathrm{id}, \mathbf{t}_b^x, \mathbf{t}_b^{x'})$ compute

$$\mathbf{t}_b^{x+x'} \leftarrow \mathbf{t}_b^x + \mathbf{t}_b^{x'} + (1 - 2b) \cdot \mathsf{PRF}(K, \mathrm{id}) \mod q.$$

- **Add input values:** On instruction $(\mathrm{id}, \mathbf{E}^x, \mathbf{E}^{x'})$ compute

$$\mathbf{E}^{x+x'} \leftarrow \mathbf{E}^x + \mathbf{E}^{x'} \mod q.$$

- **Multiply memory value by input:** On instruction $(\mathrm{id}, \mathbf{t}_b^x, \mathbf{E}^{x'})$ compute

$$\mathbf{t}_b^{x \cdot x'} := \mathsf{PKE.DDec}(b, \mathbf{t}_b^x, \mathbf{E}^{x'}) + (1 - 2b) \cdot \mathsf{PRF}(K, \mathrm{id}) \mod q.$$

- **Output from memory, as element in R_β:** On instruction $(\mathrm{id}, \mathbf{t}_b^x)$ parse $\mathbf{t}_b^x =: (x_b, \widehat{\mathbf{t}}_b^x)$ for some $x_b \in R_q$, $\widehat{\mathbf{t}}_b^x \in R_q^{d-1}$ and output

$$x_b \mod \beta.$$

Figure 5.3: 2-party public-key homomorphic secret sharing scheme HSS for the class of RMS programs from encryption with nearly linear decryption. Here, $x \in R$ with $\|x\|_\infty \leq B_{\mathrm{inp}}$ is an input value. Throughout, *input values* $x \in R$ are represented by encryptions $\mathbf{E}^x$ of $x \cdot \mathbf{s}$ and *memory values* $x \in R$ are represented by shares $(\mathbf{t}_0^x, \mathbf{t}_1^x) \in R_q^d \times R_q^d$ with $\mathbf{t}_0^x + \mathbf{t}_1^x = x \cdot \mathbf{s} \mod q$.

$\mathbf{G}_0$:	$\mathbf{G}_1$:
$(b, x_0, x_1, \mathsf{state}) \leftarrow \mathcal{A}(1^\lambda)$	$(b, x_0, x_1, \mathsf{state}) \leftarrow \mathcal{A}(1^\lambda)$
$\beta \leftarrow \{0,1\}$	$\beta \leftarrow \{0,1\}$
$(\mathsf{pk}, (\mathsf{ek}_0, \mathsf{ek}_1)) \leftarrow \mathsf{HSS.Gen}(1^\lambda)$	$(\mathsf{pk}, (\mathsf{ek}_0, \mathsf{ek}_1)) \leftarrow \mathsf{HSS.Gen}(1^\lambda)$
//Encrypt $x_\beta \cdot \mathbf{s}$.	//Encrypt $\mathbf{0} \in R^d$.
$\mathbf{E} \leftarrow \mathsf{PKE.OKDM}(\mathsf{pk}, x_\beta)$	$\mathbf{E} \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}, \mathbf{0})$
$\beta' \leftarrow \mathcal{A}(\mathsf{state}, \mathsf{pk}, \mathsf{ek}_b, \mathbf{E})$	$\beta' \leftarrow \mathcal{A}(\mathsf{state}, \mathsf{pk}, \mathsf{ek}_b, \mathbf{E})$
if $\beta' = \beta$ **return** 1	**if** $\beta' = \beta$ **return** 1
else return 0	**else return** 0

Figure 5.4: Games $\mathbf{G}_0$ and $\mathbf{G}_1$ in the proof of Theorem 57 (Sec. of HSS).

distribution of game $\mathbf{G}_0$. On the other hand, if the experiment returns encryptions of 0, the distribution of ek_b equals the distribution of game $\mathbf{G}_1$. We have thus

$$|\Pr[\mathbf{G}_0 = 1] - \Pr[\mathbf{G}_1 = 1]| \leq \mathrm{Adv}^{\mathrm{kdm-ind}}_{\mathsf{PKE.OKDM},\mathcal{B}}(\lambda).$$

As in game $\mathbf{G}_1$ the view of $\mathcal{A}$ is independent of β, it holds

$$\Pr[\mathbf{G}_1 = 1] = 1/2.$$

□

Lemma 58 (Correctness of the HSS)**.** *Let* HSS *be the HSS from Figure 5.3 with underlying ring $R = \mathbb{Z}[X]/(X^N+1)$. Then, for all $\lambda \in \mathbb{N}$, for all inputs $x^{(1)}, \ldots, x^{(\rho)} \in [R]_{B_{\mathrm{inp}}}$, for all RMS programs P, s.t.*

- *P is of size $|P| \leq \mathsf{poly}(\lambda)$*
- *P has magnitude bound $B_{\max}$ with $r/B_{\max} \geq \lambda^{\omega(1)}$ and $q/(B_{\max} \cdot r) \geq \lambda^{\omega(1)}$,*
- *P has maximum number of input addition instructions $P_{\mathrm{inp}+}$*

for $(\mathsf{pk}, \mathsf{ek}_0, \mathsf{ek}_1) \leftarrow \mathsf{HSS.Gen}(1^\lambda)$, for $\mathbf{E}^{x^{(i)}} \leftarrow \mathsf{HSS.Enc}(1^\lambda, \mathsf{pk}, x^{(i)})$, there exists an PPT adversary $\mathcal{B}$ on the pseudorandom function PRF *with such that correctness holds with probability at least*

$$\begin{aligned}\Pr^{\mathrm{cor}}_{\mathsf{HSS},(x^{(i)})_i,P}(\lambda) \geq 1 &- \mathrm{Adv}^{\mathrm{prf}}_{\mathsf{PRF},\mathcal{B}}(\lambda) - N \cdot (B_{\max}+1)/q \\ &- |P| \cdot d \cdot N^2 \cdot P_{\mathrm{inp}+} \cdot B_{\max} \cdot (B_{\mathrm{ct}} \cdot r/q + B_{\mathrm{sk}}/r)\,. \\ &- |P| \cdot d \cdot N \cdot (r/q + 1/r).\end{aligned}$$

Proof. We prove correctness via a hybrid argument. Let $\varepsilon_0 := \Pr^{\mathsf{cor}}_{\mathsf{HSS},(x^{(i)})_i,P,\beta}(\lambda)$. Recall that by ε^0 we denote the probability that homomorphic evaluation of a program P on input $(x^{(1)}, \ldots, x^{(\rho)}) \in [R]^\rho_{B_{\mathrm{inp}}}$ employing our HSS presented in Figure 5.3 is successful (over the random choices of $\mathsf{HSS.Gen}, \mathsf{HSS.Enc}$). Our goal is to prove that for all $x^{(1)}, \ldots, x^{(\rho)} \in [R]_{B_{\mathrm{inp}}}$ and for all bounded RMS programs P the probability ε_0 is negligible in λ.

To this end, let $\varepsilon_1 := \Pr^1_{\mathsf{HSS},(x^{(i)})_i,P,\beta}(\lambda)$ denote the probability that evaluation yields the correct output, where we replace every evaluation of the PRF by inserting

a value $\mathbf{r} \leftarrow_R R_q^d$ chosen at random. We show that if the probabilities ε_0 and ε_1 differ significantly, then there exists an adversary $\mathcal{B}$ attacking the underlying PRF PRF. Namely, $\mathcal{B}$ homomorphically evaluates the program P on input $(x^{(1)}, \ldots, x^{(\rho)})$, but instead of evaluating $\mathsf{PRF}(K, \mathrm{id})$ the adversary $\mathcal{B}$ queries its PRF oracle. Finally, $\mathcal{B}$ returns *real* if homomorphic evaluation does not yield the correct result, and *random* otherwise. This yields

$$|\varepsilon_0 - \varepsilon_1| \leq \mathrm{Adv}^{\mathrm{prf}}_{\mathsf{PRF},\mathcal{B}}(\lambda).$$

It is left to give a lower bound for the probability ε_1. To that end, we prove that with overwhelming probability over the choice of $\mathbf{r} \leftarrow R_q^d$ (in place of the PRF evaluation) all shares $(\mathbf{t}_0^x, \mathbf{t}_1^x)$ computed during homomorphic evaluation of P satisfy

$$\mathbf{t}_0^x + \mathbf{t}_1^x = x \cdot \mathbf{s} = (x, x \cdot \widehat{\mathbf{s}}) \mod q \tag{5.2}$$

if the function evaluation of P at point $(\mathbf{t}_0^x, \mathbf{t}_1^x)$ corresponds to $x \in R$, where $\mathbf{s} = (1, \widehat{\mathbf{s}}) \in R \times R^{d-1}$ is the secret key returned by $\mathsf{PKE.Gen}$ on input 1^λ. Further, we have that $(\mathbf{t}_0^x, \mathbf{t}_1^x)$ are distributed uniformly at random conditioned on Equation 5.2.

Assuming Equation 5.2 is true, by Lemma 54 we have $x_0 + x_1 = x$ over R (and thus over R_β) with probability at least $1 - N \cdot (B_{\max} + 1)/q$.

It is left to prove that indeed Equation 5.2 holds true during homomorphic evaluation of P except with negligible probability. Recall that $\mathsf{PKE.DDec}$ is the procedure for distributed decryption from Lemma 55. First, assume that distributed decryption is always successful. In this case we prove that any instruction preserves correctness. Note that we do not need to consider the addition of input values and the output of a memory value, as those do not affect the shares.

- **Load an input into memory:** Consider intruction $(\mathrm{id}, \mathbf{E}^x)$ for $b \in \{0, 1\}$.

  Assuming correctness of distributed decryption it holds

$$\begin{aligned}\mathbf{t}_0^x + \mathbf{t}_1^x =& \mathsf{PKE.DDec}(0, \mathbf{s}_0, \mathbf{E}^x) + \mathbf{r} + \mathsf{PKE.DDec}(1, \mathbf{s}_1, \mathbf{E}^x) - \mathbf{r} \mod q \\ =& 1 \cdot (x \cdot \mathbf{s}) \mod q = x \cdot \mathbf{s} \mod q.\end{aligned}$$

- **Add values in memory:** Assuming correctness holds for shares $(\mathbf{t}_0^x, \mathbf{t}_1^x)$ and $(\mathbf{t}_0^{x'}, \mathbf{t}_1^{x'})$ we have, as required,

$$\begin{aligned}\mathbf{t}_0^{x+x'} + \mathbf{t}_1^{x+x'} &= \mathbf{t}_0^x + \mathbf{t}_0^{x'} + \mathbf{r} + \mathbf{t}_1^x + \mathbf{t}_1^{x'} - \mathbf{r} \mod q \\ &= x \cdot \mathbf{s} + x' \cdot \mathbf{s} \mod q = (x + x') \cdot \mathbf{s} \mod q.\end{aligned}$$

- **Multiply memory value by input:** Assuming correctness holds for the share $(\mathbf{t}_0^x, \mathbf{t}_1^x)$ and assuming correctness of distributed decryption it holds

$$\begin{aligned}\mathbf{t}_0^{x \cdot x'} + \mathbf{t}_1^{x \cdot x'} =& \mathsf{PKE.DDec}(0, \mathbf{t}_0^x, \mathbf{E}^{x'}) + \mathsf{PKE.DDec}(1, \mathbf{t}_1^x, \mathbf{E}^{x'}) \mod q \\ =& x \cdot (x' \cdot \mathbf{s}) \mod q = (x \cdot x') \cdot \mathbf{s} \mod q.\end{aligned}$$

  As $\mathbf{r}$ is chosen at random, the distribution of $(\mathbf{t}_0^y, \mathbf{t}_1^y) \in R_q^d$ for $y \in \{x, x + x', x \cdot x'\}$ is random conditioned on Equation 5.2.

It is left to bound the probability that distributed decryption fails. As for all x computed throughout the evaluation of program P the distribution of $(\mathbf{t}_0^x, \mathbf{t}_1^x) \in R_q^d$ is random conditioned on Equation 5.2, by Lemma 55 for all messages $m_1 \ldots, m_{P_{\text{inp+}}} \in R_r$ and for all encryptions $\mathbf{c}_i$ of m_i that are output of $\mathsf{PKE.OKDM}$ distributed decryption of $\sum_{i=1}^{P_{\text{inp+}}} \mathbf{c}_i$ fails with probability at most

$$N^2 \cdot P_{\text{inp+}} \cdot \|x\|_\infty \cdot B_{\text{ct}} \cdot r/q + N \cdot \|x \cdot m\|_\infty / r + N \cdot (r/q + 1/r),$$

where $m := \sum_{i=1}^{P_{\text{inp+}}} m_i$. Throughout the evaluation of P we are guaranteed $\|x\|_\infty \leq B_{\max}$ for all intermediary values $x \in R$. Further, for the messages $m_i = x_i \cdot s_{j_i}$ corresponding to outputs of $\mathsf{PKE.OKDM}$ we have

$$\|x \cdot \sum_{i=1}^{P_{\text{inp+}}} x_i \cdot s_{j_i}\|_\infty \leq \sum_{i=1}^{P_{\text{inp+}}} \|x \cdot x_i \cdot s_{j_i}\|_\infty \leq P_{\text{inp+}} \cdot N \cdot B_{\max} \cdot B_{\text{sk}}.$$

Finally, applying a union bound over all $|P| \cdot d$ decryptions yields

$$\begin{aligned}\varepsilon_1 \geq 1 &- N \cdot (B_{\max} + 1)/q - |P| \cdot d \cdot N^2 \cdot P_{\text{inp+}} \cdot B_{\max} \cdot (B_{\text{ct}} \cdot r/q + B_{\text{sk}}/r) \\ &- |P| \cdot d \cdot N \cdot (r/q + 1/r).\end{aligned}$$

□

5.5 Extensions

In the following we briefly describe some extensions which are tailored to special applications and improve the HSS construction introduced in the previous section in terms of efficiency. For a complete treatment, we refer the reader to the full version of [BKS19]. For an overview of key sizes and sizes of ciphertext/shares of our schemes, we refer to Table 5.1. For an overview of the evaluation costs, we refer to Table 5.2.

Secret-Key HSS. For certain applications, where all secret inputs originate from a single party, it is sufficient to consider a *secret-key* HSS. This allows a more efficient instantiation for two reasons. First, the underlying encryption scheme is not required to support ciphertexts from a KDM oracle (but has to be KDM secure), which slightly saves in noise parameters. Further, we can save in terms of computations (at the cost of a larger share size), by replacing the DDec steps for loading an input x into memory, by instead sending the secret shares of $x \cdot \mathbf{s}$ as an additional part of the HSS share.

HSS for Degree-2 Polynomials. For the restricted class of degree-2 polynomials, we can achieve improved efficiency in both the secret-key and public-key setting, by leveraging the fact that our HSS need only support terminal multiplications.

For the secret-key case, as we do not need to load inputs, we actually only need one level of distributed decryption. This has two advantages: First, it suffices to encrypt $x \in R_r$ instead of $x \cdot \mathbf{s} \in R_r^d$, as the output is not required to allow another distributed encryption. Second, for the same reason, we do not need to lift the modulus of the output of the distributed decryption back to q. Thus, we can choose $r \leq \mathsf{poly}(\lambda)$ and $q \geq \lambda^{\omega(1)}$ (as we no longer must apply Lemma 54).

The idea of our public-key HSS is to change the way inputs are loaded into memory. The idea is to obtain the shares of $x \cdot \mathbf{s} = (x, x \cdot s_2, \ldots, x \cdot s_d) \in R^d$ by decrypting

	HSS (Fig. 5.3)	skHSS	HSS′	skHSS′
pk/sk:	$\lvert\mathsf{pk}\rvert$	d	$\lvert\mathsf{pk}\rvert$	d
ek_b:	$d + \lvert K\rvert$	$d + \lvert K\rvert$	$d^2 + \lvert K\rvert$	$d + \lvert K\rvert$
$\mathsf{ct}/\mathsf{sh}_b$:	d^2	$d^2 + d$	d	$2d$

Table 5.1: Overview of sizes of keys and of ciphertexts/shares. By HSS and skHSS we denote our constructions of public-key and secret-key HSS for the class of all RMS programs, respectively. By HSS′ and skHSS′ we denote our constructions of public-key and secret-key HSS for the class of degree-2 polynomials . Further, $\lvert\mathsf{pk}\rvert$ denotes public key size of the underlying encryption scheme, $\lvert K\rvert$ denotes PRF key size. Values are expressed in units of R_q elements. Recall that d denotes the ciphertext dimension (i.e. $d = 2$ for RLWE).

	HSS (Fig. 5.3)	skHSS	HSS′	skHSS′
Load	$d^2 \cdot \mathsf{mult}_q$	0	$d^2 \cdot \mathsf{mult}_q$	0
Add (mem,nt)	$d \cdot \mathsf{add}_q$	$d \cdot \mathsf{add}_q$	$d \cdot \mathsf{add}_q$	$d \cdot \mathsf{add}_q$
Add (mem,t)	$1 \cdot \mathsf{add}_\beta$	$1 \cdot \mathsf{add}_\beta$	$1 \cdot \mathsf{add}_\beta$	$1 \cdot \mathsf{add}_\beta$
Add (input)	$d^2 \cdot \mathsf{add}_q$	$d^2 \cdot \mathsf{add}_q$	$d \cdot \mathsf{add}_q$	$d \cdot \mathsf{add}_q$
Multiply (nt)	$d^2 \cdot \mathsf{mult}_q$	$d^2 \cdot \mathsf{mult}_q$	–	–
Multiply (t)	$d \cdot \mathsf{mult}_q$	$d \cdot \mathsf{mult}_q$	$d \cdot \mathsf{mult}_q$	$d \cdot \mathsf{mult}_q$

Table 5.2: Overview of evaluation costs, where we restrict to the dominant cost and omit the cost for evaluating the PRF. For $\mathsf{mod} \in \{\beta, q\}$ by $\mathsf{add}_{\mathsf{mod}}$ and $\mathsf{mult}_{\mathsf{mod}}$ we denote the number of additions and multiplications over R_{mod} required, respectively. By "nt" and "t" we denote *non-terminal* and *terminal* operations (i.e. not followed by another multiplication). Recall that d is the size of a ciphertext of PKE/SKE. Further, recall that for skHSS′ we can allow $r \leq \mathsf{poly}(\lambda)$ and therefore also smaller (but still super-polynomial) modulus q.

$\mathsf{PKE.Enc}(\mathsf{pk}, x)$ with $\mathbf{s}$ *and* with $s_2 \cdot \mathbf{s}, \ldots, s_d \cdot \mathbf{s}$. This strategy requires a quadratic number of secret shares (namely shares of $\mathbf{s} \cdot \mathbf{s}^\top$), but reduces the number of required encryption from d to 1 (as only encryptions of x are required). An additional advantage of this approach is that we only have to require the underlying encryption scheme to be IND-CPA secure (instead of satisfying pseudorandomness of ciphertexts).

HSS Supporting SIMD Operations. As first observed by [SV14], if the underlying ring R is of the right form, one can "pack" multiple plaintexts in one ciphertext. We show that our basic HSS supports "single instruction, multiple input" (SIMD) in this case. More precisely, we show that if $R = \mathbb{Z}[X]/(X^N + 1)$ for $N \in \mathbb{N}$, $N \leq \mathsf{poly}(\lambda)$ a power of 2, such that $X^N + 1$ splits over R_p (for some prime $p \geq 2$) into pairwise different irreducible polynomials of degree $k \in \mathbb{N}$ (i.e. $R_p \cong \left(\mathbb{F}_{p^k}\right)^{N/k}$), one can evaluate a program P simultaneously on N/k inputs in $\mathbb{F}_{p^k}$. However, there are some caveats regarding magnitude growth with respect to the SIMD versus coefficient representation.

$\underline{\mathsf{LPR.Gen}(1^\lambda):}$

1. Sample $a \leftarrow R_q, \widehat{s} \leftarrow \mathcal{D}_{\mathsf{sk}}, e \leftarrow \mathcal{D}_{\mathrm{err}}$ and compute $b = a \cdot \widehat{s} + e$ in R_q.
2. Let $\mathbf{s} = (1, \widehat{s})$ and output $\mathsf{pk} = (a, b), \mathsf{sk} = \mathbf{s}$.

$\underline{\mathsf{LPR.Enc}(\mathsf{pk}, m):}$

1. To encrypt $m \in R_r$, first sample $v \leftarrow \mathcal{D}_{\mathsf{sk}}, e_0, e_1 \leftarrow \mathcal{D}_{\mathrm{err}}$.
2. Output the ciphertext $(c_0, c_1) \in R_q^2$, where $c_1 = -av + e_0$ and $c_0 = bv + e_1 + (q/r) \cdot m$.

$\underline{\mathsf{LPR.OKDM}(\mathsf{pk}, m):}$

1. Compute $\mathbf{c}^0 = \mathsf{LPR.Enc}(0)$ and $\mathbf{c}^m = \mathsf{LPR.Enc}(m)$.
2. Output the tuple $(\mathbf{c}^m, \mathbf{c}^0 + (0, (q/r) \cdot m))$ as encryptions of $m \cdot \mathbf{s}$.

$\underline{\mathsf{LPR.DDec}(b, \mathbf{t}_b, \mathbf{c}^x):}$

1. Given $b \in \{0, 1\}$, a ciphertext $\mathbf{c}^x$ and a share $\mathbf{t}_b$ of $m \cdot \mathbf{s}$, first parse $\mathbf{c}^x = (c_0, c_1)$ and $\mathbf{t}_b = (t_{b,0}, t_{b,1})$.
2. Output $(d_0, d_1) := (\lfloor (r/q) \cdot (c_0 \cdot t_{b,0} + c_1 \cdot t_{b,1}) \rceil \mod r) \mod q$

Figure 5.5: Ring-LWE based instantiation of PKE with nearly linear decryption, with procedures for HSS from Section 5

5.6 Instantiations and Efficiency Analysis

Our HSS schemes can be instantiated in a number of ways, using LWE or RLWE-based encryption schemes satisfying the nearly-linear decryption property from Definition 50. In this section we focus on a particularly efficient RLWE-based instantiation using a variant of the "LPR" encryption scheme [LPR13] over 2-power cyclotomic rings.

In Figure 5.5 we present the core algorithms for our RLWE-based instantation using the LPR [LPR13] public-key encryption scheme $\mathsf{LPR} = (\mathsf{LPR.Gen}, \mathsf{LPR.Enc})$, as well as the auxiliary algorithms LPR.OKDM and LPR.DDec used by our HSS constructions. We use an error distribution $\mathcal{D}_{\mathrm{err}}$ where each coefficient is a rounded Gaussian with parameter σ, which gives $B_{\mathrm{err}} = 8\sigma$ as a high-probability bound on the ℓ_∞ norm of samples from $\mathcal{D}_{\mathsf{sk}}$, with failure probability $\mathtt{erf}(8/\sqrt{2}) \approx 2^{-49}$. We choose the secret-key distribution such that each coefficient of s is uniform in $\{0, \pm 1\}$, subject to the constraint that only h_{sk} coefficients are non-zero.[3]

The following lemma shows that LPR satisfies the nearly-linear decryption prop-

[3] Choosing a sparse secret like this does incur a small loss in security, and only gives us a small gain in parameters for the HSS. The main reason we choose s like this is to allow a fair comparison with SHE schemes, which typically have to use sparse secrets to obtain reasonable parameters.

erty for our HSS scheme. Furthermore, notice that ciphertexts output by LPR.Enc are pseudorandom under the decisional ring-LWE assumption, by a standard hybrid argument [LPR10]. Therefore, the correctness and security properties of the LPR.OKDM and LPR.DDec procedures follow from Lemmas 52 and 55.

Lemma 59. *Assuming hardness of* $\mathsf{RLWE}_{N,q,\mathcal{D}_{\mathrm{err}},\mathcal{D}_{\mathrm{sk}}}$*, the scheme* LPR *(Figure 5.5) is a public-key encryption scheme with nearly-linear decryption over* $R = \mathbb{Z}[X]/(X^N + 1)$*, with ciphertext dimension* $d = 2$ *and bounds* B_{sk} *and* $B_{\mathrm{ct}} = B_{\mathrm{err}} \cdot (2h_{\mathrm{sk}} + 1)$*.*

Proof. Notice that for a ciphertext $\mathbf{c} = (c_0, c_1) = \mathsf{LPR.Enc}(m)$, we have

$$\langle \mathbf{c}, \mathbf{s} \rangle = c_0 + c_1 \cdot \widehat{s} = bv + e_1 + (q/p) \cdot m - avs + e_0\widehat{s} = ev + e_1 + e_0 s + (q/p) \cdot m$$

This means that LPR satisfies property 3 of Definition 50, with noise bound B_{ct} given by the maximum of $\|ev + e_1 + e_0 s\|_\infty$. Since $\|e\|_\infty \leq B_{\mathrm{err}}$ and v has only h_{sk} non-zero coefficients of ± 1, it follows that the product ev has coefficients bounded by $B_{\mathrm{err}} \cdot h_{\mathrm{sk}}$. Summing up, the total noise bound in a fresh ciphertext is therefore $B_{\mathrm{ct}} = B_{\mathrm{err}} \cdot (2h_{\mathrm{sk}} + 1)$.

We conclude that LPR is a PKE with approximately linear decryption (Definition 50) and parameters $(p, q, d = 2, B_{\mathrm{sk}} = 1, B_{\mathrm{ct}}, R)$. □

Parameters and Efficiency Analysis. We now analyse the efficiency of our RLWE-based instantiation and compare it with using HSS constructed from somewhat homomorphic encryption, for various different settings of parameters.

For comparison with HSS based on DDH [BGI16a], we remark that for non-SIMD computations, DDH-based HSS shares can be smaller than both our approach and SHE. However, we estimate that homomorphic evaluation is around an order or magnitude faster than the times reported in [BCG+17] due to the expensive share conversion procedure, and when using SIMD both this and the share size can be dramatically improved.

Parameter Estimation. We derived parameters for our HSS based on LPR using the bounds for correctness from Lemma 58, chosen to ensure that each RMS multiplication of a ring-element during evaluation is correct with probability $1 - 2^{-\kappa}$, where we chose $\kappa = 40$. To compare with constructing HSS from SHE, we estimated parameters for the "BFV" scheme based on RLWE [Bra12, FV12], currently one of the leading candidate SHE schemes. To modify this to achieve HSS with additive output sharing, we need to increase the size of q by around 2^κ bits. With both schemes we chose parameters estimated to have at least 80 bits of computational security, see the full version of [BKS19] for more details.

Share Size. Tables 5.3–5.4 show BFV ciphertext parameters for different multiplicative depths of circuit, and plaintext modulus 2 or $\approx 2^{128}$, respectively, to illustrate different kinds of Boolean and arithmetic computations. Table 5.5 gives our HSS parameters for various choices of $B_{\max}$, the maximum value any plaintext coefficient can hold during the computation. Note that in contrast to SHE, our parameters depend only on this bound and not the multiplicative depth, although we are more restricted in that we can only perform homomorphic multiplications where one value is an input.

Depth	N	$\log q$	Security
1	4096	102	145.1
2	4096	118	122.6
3	4096	134	106.2
4	4096	150	93.73
5	4096	164	85.53
6	8192	186	157.5
7	8192	202	142.9
8	8192	220	129.8
9	8192	236	120.1
10	8192	252	111.9

Table 5.3: BFV with $r = 2$

Depth	N	$\log q$	Security
1	16384	456	124.3
2	16384	602	92.44
3	32768	750	154.2

Table 5.4: BFV with $r \approx 2^{128}$

$B_{\max}$	N	$\log q$	Security
2	4096	137	103.3
2^{16}	4096	167	83.74
2^{32}	8192	203	142.0
2^{64}	8192	267	104.9
2^{128}	16384	399	143.9
2^{256}	16384	655	84.60

Table 5.5: RLWE based HSS for RMS programs w/ max plaintext size $B_{\max}$

This means that comparing parameters of the two schemes is very application-dependent. For instance, for Boolean computations where we can have $B_{\max} = 2$, our scheme has smaller parameters than SHE for all computations of depth > 3, so this can give a significant advantage for very high degree functions that can be expressed as an RMS program. However, if SIMD computations are required then $B_{\max}$ must be chosen to account for the worst-case coefficient growth, which is not directly related to the plaintexts, so our scheme would likely have larger ciphertexts than SHE in most cases. For operations on large integers, the parameters in both schemes quickly get very large, though our parameters grow slightly quicker due to the increase in $B_{\max}$.

Computational efficiency. The relative computational efficiency of the schemes is much clearer, and is the main advantage of our scheme over SHE. The cost of a homomorphic RMS multiplication with RLWE is roughly twice the cost of a decryption in any RLWE-based scheme (including BFV) with the same parameters. Recently, Halevi et al. [HPS18] described an optimized implementation of BFV using CRT arithmetic, where according to their single-threaded runtimes, decryption costs between 20–30x less than multiplication (including key-switching) for the ranges of parameters we consider (cf. [HPS18, Table 3]). This indicates a *10–15x improvement in performance* for homomorphic evaluation with our scheme compared with SHE, assuming similar parameters and numbers of multiplications. We remark that this comparison deserves some caution, since other SHE schemes such as BGV [BGV12] may have different characteristics; we have not run experiments with BGV, but due to the complications in key-switching and modulus-switching we expect the improvement to still be around an order of magnitude.

5.7 Applications

In this section we highlight some applications of HSS for which our scheme seems well-suited. There are four primary approaches to compare: approaches not relying on HSS, using DDH-based or one-way function-based HSS, using HSS based on SHE, or using our new HSS. We remark that the concrete practicality of SHE-based HSS approaches has also not been considered before this work.

Secure 2-PC for Low-Degree Polynomials. Perhaps the most natural appli-

cation of HSS is to achieve a very succinct form of multi-party computation. After a setup phase to create the key material $\mathsf{pk}, (\mathsf{ek}_0, \mathsf{ek}_1)$, each party publishes HSS-shares of its input, which can then be directly used to compute additive shares of the output. Even the simplest case of evaluating degree-2 polynomials has many interesting applications, and also allows us to use our optimized HSS scheme from Section 5.5, where shares consist of *a single* RLWE ciphertext, instead of two. The main motivating example we look at is to MPC protocols in the *preprocessing model*, where correlated randomness is pre-generated ahead of time to help increase efficiency when the actual computation takes place. This correlated randomness can take many forms, but the most common are Beaver triples, namely additive shares of (a, b, c) where $c = a \cdot b$ and a, b are random elements of a (typically) large prime field. These can easily be generated using degree-2 HSS, where each party inputs two field elements, and are also highly amenable to SIMD processing.

Looking at Tables 5.4–5.5, for an example of degree 2 functions over a 128-bit message space, BFV with depth 1 requires a dimension $N = 16384$ and modulus $\log q = 456$, whereas our scheme would need to use $B_{\max} \approx 2^{256}$, giving the same dimension and a slightly larger modulus of around 655 bits. Therefore, our communication cost will be slightly larger than using SHE-based HSS, but we expect to gain from the lower computational costs that come with our multiplication.

Using DDH-type HSS [BCG$^+$17], an m-bit triple can be created with $3712(5m/4+160)$ bits of communication, giving 148kB for $m = 128$, meaning our communication is 20x higher for producing a single triple (at 2682kB), but orders of magnitude smaller ($\sim$900x) when amortized using SIMD (over $N = 16834$ triples). Computation requirements will greatly favor our approach.

We can also compare this with other approaches to Beaver triple generation. The SPDZ protocol [DPSZ12] uses SHE (without HSS) to create triples; as well as the more complex homomorphic multiplication, this incurs extra costs in an interactive distributed decryption protocol, which adds a round of interaction that we can avoid using HSS with local rounding. The latest version of SPDZ [KPR18] uses linearly-homomorphic encryption instead of SHE, and reports ciphertexts with $\log q$ as small as 327 bits, around half the size of ours. This would likely beat HSS in terms of communication and computation, but still has the undesirable feature of 2 rounds of interaction, whereas with HSS (and a small one-time setup), the triples are obtained after just one message from each party.

For an approach with communication *sublinear* in the number of triples, we refer Chapter 6 in this thesis. Building on our HSS presented in this chapter (instantiated with the BGV encryption scheme [BGV12]), this allows to generate 17 GB of Beaver triples over $\mathbb{F}_p$ for $p \approx 2^{128}$ at a rate of 0.16 ms per triple with communication complexity of approximately 0.18 bit per triple-bit generated. Note that here the more general case of authenticated Beaver triples is considered. This results into (amortized) about 138 bit of communication per 128-bit triple (where each triple consist of 6 values in $\mathbb{F}_p$). For more details we refer to Section 6.5.

2-Server (Generalized) Private Information Retrieval. An attractive application of HSS is to obtain highly succinct Private Information Retrieval (PIR) protocols for $m \geq 2$ servers. Here, m servers hold a public database DB and allow clients to submit private queries to DB, such that both the query and response re-

main hidden to up to $m-1$ colluding servers.[4] When using HSS, we can obtain a very simple, 1-round protocol where the client first sends an encryption of its query to both servers, who respond with an additive share of the result. Note that we only need the more efficient, secret-key version of HSS, such as our scheme from Section 5.5 with $m = 2$ servers.

Recent works on 2-server PIR have used HSS for point functions[5] to support basic queries including equality conditions, range queries and disjoint OR clauses, based on simple schemes using only one-way functions [BGI16b, WYG+17]. However these techniques degrade dramatically for more complex queries, due to the relatively weak homomorphic ability of the underlying HSS. With HSS for branching programs we can significantly increase the expressiveness of queries, at the cost of some overhead in ciphertext size and running time.

In a bit more detail, suppose that a client issues a simple $COUNT$ query,[6] which applies some predicate Q to each row $x_i \in \mathsf{DB}$, and returns $\sum_i Q(x_i)$, that is, the number of rows in DB that match Q. The general idea is that the client splits Q into HSS shares s^1, s^2, and sends s^j to server j. For each row $x_i \in \mathsf{DB}$, the servers then use homomorphic evaluation with the function $f_{x_i}(Q) := Q(x_i)$ on the shares, to obtain a shared 0/1 value indicating whether a match occurred. Given additive shares modulo β of the results $q_1, \ldots, q_D$ (where $D = |\mathsf{DB}|$), the servers can sum up the shares and send the result to the client, who reconstructs the result $q = \sum q_i$ (this assumes that $\beta < N$, so wraparound does not occur).

Below we analyse some useful classes of predicates that are much more expressive than function classes that can be handled using one-way function based approaches, and seem well-suited for our scheme supporting RMS programs.

Conjunctive Keyword Search. Suppose that each entry in DB is a document x with a list of keywords $W_x = \{w_1^x, \ldots, w_m^x\}$, and the query is a $COUNT$ query consisting of an arbitrary conjunction of keywords, each in $\{0,1\}^\ell$. That is, for a query $W = \{w_1, \ldots, w_k\}$ containing keywords shared bit-by-bit using the HSS, the servers will compute a sharing of

$$\#\{(x, W_x) \in \mathsf{DB} : W \subseteq W_x\}$$

To evaluate the query on a single entry of DB as an RMS program, we maintain the result f as a secret-shared memory value, which is initially set to 1. We then iterate over each query keyword $w_i \in W$, letting w_{ij} denote the j-th bit of w_i, and update f as

$$f := \sum_{w^x \in W_x} f \cdot \prod_{j=1}^{m} (1 \oplus w_j^x \oplus w_{ij})$$

Note that the i-th product evaluates to 1 iff $x^x = w_i$, and since all w^x are distinct, at most one of these will be 1. Multiplication by f applies a conjunction with the

[4] Using S/FHE alone instead of HSS allows for the stronger setting of single-server PIR. However, a major advantage of HSS with additive reconstruction is that shares across many rows can easily be combined, allowing more expressive queries with simpler computation.

[5] Actually, these works use *function secret-sharing* [BGI15] for point functions, which in this case is equivalent to HSS for the same class of functions.

[6] Other queries such as returning the record identifier, or min/max and range queries can easily be supported with similar techniques, as previously shown in [WYG+17, BCG+17].

previous keyword, and must be performed inside the summation as f is a memory value. All other product terms are linear functions (over $\mathbb{Z}$) in the inputs w_i (via $a \oplus b = a + b - 2ab$), so each product can be evaluated left-to-right as an RMS program, for a total of $m \cdot \ell \cdot k$ RMS multiplications after iterating over all k query keywords.

Comparison to SHE-based HSS. When using SHE, the number of homomorphic multiplications is roughly the same as our case, and the multiplicative depth is $\log(m\ell k)$. For a concrete example, suppose that each document has $m = 10$ keywords of length $\ell = 128$ bits, and a client's query has $k = 4$ keywords. Using either our HSS scheme or HSS from SHE would need around 5120 multiplications per document, with a multiplicative depth of 13. This needs SHE parameters of $\log q \approx 300$ and dimension $N = 8192$ for the BFV scheme as above, whereas with our scheme we can use the best case of $B_{\max} = 2$, giving $\log q \approx 137$ and $N = 4096$. Using our secret-key HSS and LPR instantiation, the share size is $3N \log q$ bits $\approx$ 210kB, around 1/3 of the SHE ciphertext size using BFV. The communication cost for the whole query would be 107MB for our HSS, and 314MB with BFV, whilst we estimate the computational costs of homomorphic evaluation per document are around 2.5s and 300s, respectively, so even with the relatively high communication cost, for matching several documents using our HSS would certainly give a significant performance improvement.

However, one drawback of our approach is that handling SIMD computations is more challenging, since the $B_{\max}$ bound must be chosen much larger to account for the coefficient growth of the plaintext polynomials, which may continue to grow even when the packed plaintext messsages themselves are only bits. If the number of documents in the database is large enough to warrant SIMD processing then it seems likely that SHE will be preferable, since $N = 8192$ documents could be searched at once without increasing the parameters.

Pattern-Matching Queries. Suppose here that the client wants to search for the occurrence of a pattern $p = (p_1, \ldots, p_m) \in \{0,1\}^m$ in each row $x = (x_1, \ldots, x_N) \in \{0,1\}^N$. An RMS program for computing the pattern-matching predicate, with public input x and private input p, can be done with $m \cdot N$ multiplications using a similar method to the previous example, modified slightly to compute the OR of matching p with every position in x.

Comparison to SHE-based HSS. When using SHE, this computation has depth $\log(nm)$, also requiring around $N \cdot m$ homomorphic multiplications. The comparison with our scheme is then similar to the keyword search example, depending on the parameters chosen. For another example, if we have a fairly large string of length $N = 10000$, and a pattern of size $m = 100$, then the SHE-based HSS must support depth 20, giving parameters $(N, \log q) = (16384, 434)$. Again, we can use our HSS with parameters for $B_{\max} = 2$, which lead to ciphertexts around 8.5x smaller than with SHE.

Chapter 6

Pseudorandom Correlation Generators

In the last chapter of this thesis we take a closer look at protocols for secure computation in the preprocessing model. Here, the expensive part of the computation is sourced out into an input-independent preprocessing phase, where the parties set up long correlated random strings. The actual computation on the inputs then takes place in a much more efficient online phase.

For an example consider the correlation of Beaver triples. Given additive shares of a tuple (a, b, ab), two parties can compute an online multiplications of additively shared values x and y as follows:

- The parties jointly open $x + a$ and $y + b$.
- Note that the parties have additive shares of a, b and ab and know $x + a$ and $y + b$ in the clear. This allows to (locally) compute an additive share of $x \cdot y$ via the following equation:

  $$x \cdot y = (x + a - a) \cdot (y + b - b) = (x + a) \cdot (y + b) - (x + a) \cdot b - a \cdot (y + b) + ab.$$

As this strategy requires *one* Beaver triple per multiplication, typically during preprocessing many Beaver triples have to computed. With traditional approaches (e.g. [DPSZ12]) this requires communication complexity (and memory) of size *at least linear* in the number of Beaver triples to be produced.

To produce *real* random triples this amount of communication is inherent by an information theoretic argument. But—as for the generation of Beaver triples one have to rely on computational assumptions anyway [Bea96]—one can ask whether this is inherent. More precisely: *Can one generate a large amount of Beaver triples with communication complexity sublinear in the number of triples?*

In our work [BCG+19b], we answer this question affirmatively. Moreover, we are the first to achieve this at a reasonable computational efficiency.

Towards this goal, we put forward the notion of *pseudorandom correlation generator (PCG)*. A pseudorandom correlation generator allows to expand short *correlated* seeds to long *correlated* pseudorandom strings (e.g. many Beaver triples). As explained in the introduction, finding the right definition for pseudorandom correlation generators on its own is non-trivial. We justify our definition, by both ruling out a more natural notion, and showing that our definition suffices to replace the preprocessing phase by a PCG in many interesting contexts.

On the practical side, we give generic construction based on homomorphic secret sharing for additive correlations. The idea is to use the homomorphic property of the HSS as follows: Giving shares of some seed, the parties first locally expand this seed by homomorphically applying a pseudorandom generator. Subsequently, the parties

Underlying HSS	|key|	|triples|	setup	expansion	exp./triple
[BGV12]	3 GB	17 GB	≈ 20 s	8.0 h	0.16 ms
[BGV12] (iterative)	3 GB	1.6 MB/it.	≈ 20 s	10 s/it.	0.57 ms
[BGV12] (w/ packing)	6 MB	1.1 GB	< 0.1 s	900 h	280 ms
[BGV12, BKS19]	3 GB	17 GB	≈ 20 s	7.6 h	0.15 ms

Figure 6.1: Overview of estimated efficiency of lattice-based approach to generate authenticated Beaver triples over $\mathbb{F}_r$. The numbers provided are time estimates for joint seed generation (with security against semi-honest adversaries) and expansion. The numbers are based on [CS16], for [BGV12] supporting depth-4 homomorphic operations (note that depth-3 would suffice to compute a degree-5 polynomial as required) and plaintext space modulus $r \approx 2^{128}$. The runtime estimates are based on NFLLib [ABG+16] with ciphertext modulus $\log q \approx 744$ and ring dimension $N = 2^{14}$. We instantiate the MQ assumption with $n = 2^9$ and $m = 2^{18}/24$, and choose sparsity $\rho = 100$ (that is, the number of non-zero coefficients per polynomial). The number of (maximal) obtained triples is $2^{32}/24$ for the rows with naive packing (including the iterative approach) and $2^{28}/24$ for the row w/ (smart) packing. Setup requires communication of roughly size |key| per party. We ignore small contributions like setting up the public key and generating suitable shares of the MAC key α, as computation and communication are dominated by generation and distribution of encryptions of the PRG seed.

evaluate the correlation itself on the long pseudorandom string, to each obtain an additive share of the long correlated string.

PCG for Authenticated Beaver Triples. In the following we give a brief overview on how to instantiate the described generic approach for the correlation of authenticated Beaver triples, that is additive shares of

$$(a, b, ab), (a\alpha, b\alpha, ab\alpha)$$

for some fixed MAC key $\alpha \in \mathbb{Z}_r$ and values $a, b \in \mathbb{Z}_r$. The message authentication code allows to detect parties that deviate the protocol in the online phase. One example, where authenticated Beaver triples find application is the so-called SPDZ-protocol [DPSZ12].

As pseudorandom generator we use a degree-2 PRG from the *multivariate quadratic (MQ)* assumption. The MQ assumption states that it is difficult to invert a system of $m > n$ quadratic equations in n variables. In [BGP06] it is shown that for random equations that assuming the system is hard to invert implies pseudorandomness of the output. For efficiency we make use of the fact that known attacks do not significantly improve, when the MQ assumption is instantiated with a sparse matrix. (Note though that our specific choice of sparseness is somewhat arbitrary.)

The function we have to homomorphically evaluate is of degree 5 as $a = \mathsf{PRG}(r_a)$ (and similar for b), where PRG is the MQ-based PRG. For all our instantiations we build on the encryption scheme of Brakerski et al. [BGV12]: Either using their homomorphic encryption scheme directly as HSS, or using a hybrid between somewhat homomorphic encryption and our HSS from Chapter 5. Our HSS does not seem competitive on its own in this particular setting, because we have to account for the plaintext magnitude, which would lead to significantly larger parameters.

For better stretch we build on naive ciphertext packing [SV11]: Instead of encrypting a single $\mathbb{Z}_r$-element we 'pack' N $\mathbb{Z}_r$-elements into each ciphertext (here, N corresponds to the dimension of the plaintext space over $\mathbb{Z}_r$). This allows to expand $2n$ ciphertexts (each containing N plaintexts) to $N \cdot m$ shared authenticated Beaver triples. Note that an encryption of α (in each slot) is reused across all instances.

Using the sparseness of the matrix, we observe that - at slightly larger computational costs - not all triples have to be computed at once, but only N at a time, corresponding to the number of plaintexts packed in one ciphertext. This is very desirable for settings, where not all Beaver triples are needed at once.

In order to achieve (almost) true quadratic stretch, one would need to use the packing more smartly, by letting the ciphertext slots interact with each other. This indeed yields a stretch from a few megabytes to more than a gigabyte, but unfortunately is not practical, as it requires expensive key switching operations.

We provide efficiency estimates of the different approaches for joint seed generation (with security against semi-honest adversaries) and silent expansion in Figure 6.1 (Figure is taken with modifications from [BCG+19b]).

Roadmap. We start the chapter by defining pseudorandom correlation generators for general correlations in Section 6.1. In Section 6.2 we rule out a simpler and more natural simulation-based definition of PCG. In Section 6.3 we show that our definition of PCG can serve as a black-box replacement of long correlated strings in a wide range of natural and practical secure protocols. In Section 6.4 we give a high-level construction of PCGs from a homomorphic secret sharing scheme together with a suitable pseudorandom generator. Finally, in Section 6.5, we give an instantiation based on lattices for the correlation of authenticated Beaver triples.

The following is taken in large parts verbatim from our work [BCG+19b].

6.1 Defining Pseudorandom Correlation Generators

At a high level, a pseudorandom correlation generator (PCG) for some relation takes as input a pair of short, correlated seeds and outputs long correlated pseudorandom strings, where the expansion procedure is deterministic and can be applied locally.

For correctness we require that the expanded output of a PCG is indistinguishable from truly random correlated strings.

For security it would be natural and straightforward to require that we can securely replace long correlated strings by short correlated seeds in any secure protocol execution. Unfortunately, as shown in the following section, this security requirement would be impossible to meet. Therefore, we will introduce (and subsequently prove useful) an indistinguishability based security notion. Namely, we require that an adversary given access to one of the short seeds k_σ, cannot distinguish the pseudorandom string $\mathsf{R}_{1-\sigma}$ from a pseudorandom string that is chosen at random conditioned on $(\mathsf{R}_0, \mathsf{R}_1)$ being correlated (where $\mathsf{R}_\sigma = \mathsf{PCG}(\mathsf{k}_\sigma)$). In other words, an adversary given access to a short seed cannot learn more about the other party's pseudorandom string than what is obvious given access to its own pseudorandom string.

In order to formally define pseudorandom correlations, we first introduce the concept of a *correlation generator* as a PPT algorithm outputting correlated elements.

Definition 60 (Correlation Generator)**.** A PPT algorithm $\mathcal{C}$ is called a *correlation generator*, if $\mathcal{C}$ on input 1^λ outputs a pair of elements in $\{0,1\}^n \times \{0,1\}^n$ for $n \in \mathsf{poly}(\lambda)$.

In order to define security, we require the notion of a reverse-sampleable correlation generator introduced in the following.

Definition 61 (Reverse-sampleable Correlation Generator)**.** Let $\mathcal{C}$ be a correlation generator. We say $\mathcal{C}$ is *reverse sampleable* if there exists a PPT algorithm $\mathsf{RSample}$ such that for $\sigma \in \{0,1\}$ the correlation obtained via:

$$\{(\mathsf{R}'_0, \mathsf{R}'_1) \,|(\mathsf{R}_0, \mathsf{R}_1) \leftarrow_R \mathcal{C}(1^\lambda), \mathsf{R}'_\sigma := \mathsf{R}_\sigma, \mathsf{R}'_{1-\sigma} \leftarrow_R \mathsf{RSample}(\sigma, \mathsf{R}_\sigma)\}$$

is computationally indistinguishable from $\mathcal{C}(1^\lambda)$.

The following definition of pseudorandom correlation generators can be viewed as a generalization of the definition of the pseudorandom VOLE generator in [BCGI18]. Note though that we do not enforce perfect correctness.

Definition 62 (Pseudorandom Correlation Generator (PCG))**.** Let $\mathcal{C}$ be a reverse-sampleable correlation generator. A *pseudorandom correlation generator (PCG) for* $\mathcal{C}$ is a pair of PPT algorithms $(\mathsf{PCG.Gen}, \mathsf{PCG.Expand})$ with the following syntax:

- $\mathsf{PCG.Gen}(1^\lambda)$: On input of the security parameter λ, outputs a pair of seeds $(\mathsf{k}_0, \mathsf{k}_1)$;
- $\mathsf{PCG.Expand}(\sigma, \mathsf{k}_\sigma)$: On input of a party index $\sigma \in \{0,1\}$ and a seed k_σ, outputs a bit string $\mathsf{R}_\sigma \in \{0,1\}^n$.

Further, the algorithms $(\mathsf{PCG.Gen}, \mathsf{PCG.Expand})$ should satisfy the following:

Correctness The correlation obtained via

$$\{(\mathsf{R}_0, \mathsf{R}_1) \,|(\mathsf{k}_0, \mathsf{k}_1) \leftarrow_R \mathsf{PCG.Gen}(1^\lambda), \mathsf{R}_\sigma \leftarrow \mathsf{PCG.Expand}(\sigma, \mathsf{k}_\sigma) \text{ for } \sigma \in \{0,1\}\}$$

is computationally indistinguishable from $\mathcal{C}(1^\lambda)$.

Security For any $\sigma \in \{0,1\}$, the following two distributions are computationally indistinguishable:

$$\{(\mathsf{k}_{1-\sigma}, \mathsf{R}_\sigma) \mid (\mathsf{k}_0, \mathsf{k}_1) \leftarrow_R \mathsf{PCG.Gen}(1^\lambda), \mathsf{R}_\sigma \leftarrow \mathsf{PCG.Expand}(\sigma, \mathsf{k}_\sigma)\} \text{ and}$$
$$\{(\mathsf{k}_{1-\sigma}, \mathsf{R}_\sigma) \mid (\mathsf{k}_0, \mathsf{k}_1) \leftarrow_R \mathsf{PCG.Gen}(1^\lambda), \mathsf{R}_{1-\sigma} \leftarrow \mathsf{PCG.Expand}(\sigma, \mathsf{k}_{1-\sigma}), \mathsf{R}_\sigma \leftarrow_R \mathsf{RSample}(\sigma, \mathsf{R}_{1-\sigma})\}$$

where $\mathsf{RSample}$ is the reverse sampling algorithm for correlation $\mathcal{C}$.

Note that the above definition is trivial to achieve in general: We can let $\mathsf{PCG.Gen}$ on input 1^λ return $(\mathsf{R}_0, \mathsf{R}_1) \leftarrow \mathcal{C}(1^\lambda)$, and simply define Expand to be the identity. Typically, we will be interested in non-trivial constructions of PCGs, in which the seed size is significantly shorter than the output size. A pseudorandom generator with image in $\{0,1\}^n$ is a simple example for an expanding PCG for the equality correlation $\{(\mathsf{R}, \mathsf{R}) \mid \mathsf{R} \in \{0,1\}^n\}$. In the following we will be interested in constructing PCGs for a much broader class of correlations, like OT correlations, OLE correlations and (authenticated) Beaver triples.

6.2 Impossibility of a Simulation-Based Definition

A natural and useful alternative to the security definition in the previous definition is the following: In any secure protocol (say against semi-honest adversaries), one can replace sampling a pair of strings from the correlation $\mathcal{C}$ by generating a pair of seeds (which are later expanded) using a PCG for $\mathcal{C}$ without compromising security. Unfortunately, as sketched in [GI99], a non-trivial PCG construction cannot satisfy such a simulation-based definition. Consider the simple protocol, where P_0 samples a pair $(\mathsf{R}_0, \mathsf{R}_1) \leftarrow \mathcal{C}(1^\lambda)$ and sends R_1 to P_1, who simply outputs R_1. This protocol obviously realizes the protocol dictated by $\mathcal{C}$, with one-sided security against P_1. But, if P_0 instead generates $(\mathsf{k}_0, \mathsf{k}_1)$ according to the seed generation algorithm of the PCG and sends k_1 to P_1, a possible simulator runs into the following problem. Simulating the above protocol given only the output R_1 corresponds to finding a short seed k_1 that can be (deterministically) expanded to R_1. If the entropy in the second output of $\mathcal{C}$ exceeds the seed-length $|\mathsf{k}_1|$, such a compression violates correctness, as it could be used to distinguish R_1 from a string that is indeed chosen via $\mathcal{C}$.

In the following, we present a formal and more general version of the above argument for ruling out a simulation-based definition for non-trivial correlations. Our negative result is based on a lower bound given by Hubáček and Wichs [HW15]. There, the notion of Yao incompressibility entropy, the computational equivalent to Shannon entropy, is employed to establish a lower bound on the required communication in a secure protocol with long outputs. More precisely, Yao incompressibility entropy [HLR07, Yao82b] is a measure on how well outputs of a distribution can be compressed on average, when the compressing and decompressing algorithms are required to be efficient. For example, a pseudorandom bit string of length ℓ has Yao incompressibility entropy ℓ.

Definition 63 (Yao Incompressiblity Entropy [HLR07] (simplified))**.** Let $\ell = \ell(\lambda) \in \mathbb{N}$. A probability ensemble $X = \{X_\lambda\}$ has *Yao incompressibility entropy at least* ℓ, if for every pair of polynomial sized circuit-ensembles $C = \{C_\lambda\}$, $D = \{D_\lambda\}$ where C has output bit-length at most $\ell - 1$, there exists a negligible function $\mathsf{negl}\colon \mathbb{N} \to \mathsf{R}^+$ such that for every sufficiently large positive integer λ we have

$$\Pr\left[x \leftarrow X\colon D(C(x)) = x\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda).$$

One of the main results of [HW15] is that the communication in a secure protocol has to at least meet the Yao incompressibility entropy of the output, when the adversary is allowed to fix the random coins of the corrupted party. Applying this result rules out meaningful PCG instantiations of a simulation-based security definition.

Theorem 64 (Impossiblity of Simulation-Based Definition for Non-Trivial PCGs)**.** *Let $\mathcal{C}$ be a reverse-sampleable correlation generator, where the Yao incompressibility entropy of the output is ℓ. Then, for every pseudorandom correlation generator* $\mathsf{PCG} = (\mathsf{PCG.Gen}, \mathsf{PCG.Expand})$ *satisfying simulation-based security, the output of the seed generation* $\mathsf{PCG.Gen}$ *algorithm must at least have bit-length ℓ.*

Proof. Let $\mathsf{PCG} = (\mathsf{PCG.Gen}, \mathsf{PCG.Expand})$ be a pseudorandom correlation generator for $\mathcal{C}$ that satisfies simulation-based security. Then, in particular, the following protocol Π_{PCG} has to satisfy one-sided security against P_1: Party P_0 runs $(k_0, k_1) \leftarrow_R \mathsf{PCG.Gen}(1^\lambda)$ and sends k_1 to P_1. Finally, P_1 outputs $\mathsf{R}_1 \leftarrow \mathsf{PCG.Expand}(1, k_1)$.

Let ℓ_1 be the Yao incompressibility entropy of the output of $\mathcal{C}^1(1^\lambda) := \{\mathsf{R}_1 \mid (\mathsf{R}_0, \mathsf{R}_1) \leftarrow \mathcal{C}(1^\lambda)\}$. Further, let

$$\mathcal{C}^1_{\mathsf{PCG}}(1^\lambda) := \{\mathsf{R}_1 \mid (\mathsf{k}_0, \mathsf{k}_1) \leftarrow_R \mathsf{PCG.Gen}(1^\lambda), \mathsf{R}_1 \leftarrow \mathsf{PCG.Expand}(1, \mathsf{k}_1)\}.$$

By correctness of the PCG, the output of $\mathcal{C}^1_{\mathsf{PCG}}$ (and therefore the output of the protocol Π_{PCG}) must meet the Yao incompressibility entropy ℓ_1, as an efficient pair of compressor and decompressor could be used as a distinguisher between $\mathcal{C}^1$ and $\mathcal{C}^1_{\mathsf{PCG}}$.

By [HW15, Theorem 5], for any protocol between two parties P_0 and P_1 with one-sided security against "honest-but-deterministic"[1] P_1, where P_1 has no input, it holds: *If the Yao incompressibility entropy of the output of P_1 is ℓ_1, then the communication complexity from P_0 to P_1 must be at least ℓ_1 bits.*

Therefore, as seed expansion is deterministic, the bit-length $|k_1|$ of the seed of the second party must be at least ℓ_1. Reversing the roles of P_0 and P_1 together with additivity of Yao incompressibility entropy yields the required. □

For the special case, where $\mathcal{C}$ outputs pairs of identical random strings (R, R), Gilboa and Ishai [GI99] sketched why pseudorandom pads cannot securely substitute perfectly-random pads returned by $\mathcal{C}$ in every secure protocol. We obtain this result as a straightforward corollary of Theorem 64.

Corollary 65. *Let $\mathcal{C}$ be the correlation generator that on input 1^λ draws a string $\mathsf{R} \leftarrow_R \{0,1\}^\ell$ uniformly at random and returns (R, R). Then, there exists no pseudorandom generator with seed length strictly less than ℓ, such that sampling a seed (and later expanding via the pseudorandom generator) can securely replace sampling uniformly at random from $\{0,1\}^\ell$ in every protocol.*

The above negative result gives a general counterexample for randomized functionalities. In the full version of [BCG+19b] we show that PCGs cannot replace correlated randomness in general, even in protocols that only realize *deterministic* functionalities.

6.3 Applying PCGs in Protocols with Correlated Randomness

In this section we show that one *can* use PCGs in a "plug-and-play" fashion in protocols consuming correlated randomness sampled by a given functionality. More precisely, we show that PCGs can be directly applied to any protocol using a weaker form of correlated randomness, where corrupted parties can influence their outputs.

A simple example are random Beaver triples, where the weaker functionality we can realize allows a corrupt sender/receiver to choose its outputs, then the other party's outputs are sampled at random correspondingly. When using Beaver triples in an MPC protocol, the Beaver triples are typically used to mask the actual inputs to a multiplication. Allowing a corrupt party to choose its own share of the Beaver triple does not affect the security of these protocols, since (intuitively) this can only

[1] A "honest-but-deterministic" adversary has to behave according to the protocol, but is allowed to fix its random coins.

Functionality $\mathcal{F}^{\mathcal{C}}_{\mathsf{corr}*}$

On input 1^λ, the functionality does as follows:

- If no parties are corrupt, sample $(\mathsf{R}_0, \mathsf{R}_1) \leftarrow_R \mathcal{C}(1^\lambda)$.
- Otherwise, if P_σ is corrupt, wait to receive $\mathsf{R}_\sigma \in \{0,1\}^{n_\sigma}$ from Adv, then sample $\mathsf{R}_{1-\sigma} \leftarrow_R \mathsf{RSample}(\sigma, \mathsf{R}_\sigma)$.

The functionality outputs R_0 to P_0 and R_1 to P_1, and then halts.

Figure 6.2: Corruptible correlated randomness functionality for a reverse-sampleable correlation generator, $\mathcal{C}$

weaken security for the corrupt party and not for honest parties. More generally, it turns out that many practical MPC protocols, including those based on preprocessed multiplication triples for arithmetic circuits [BDOZ11, DPSZ12] and binary circuits [NNOB12, WRK17a, WRK17b], use this kind of corruptible, correlated randomness, since it is often easier to design a protocol that realizes this.

More formally, the randomness is modelled by the functionality $\mathcal{F}^{\mathcal{C}}_{\mathsf{corr}*}$ (Figure 6.2), where a corrupted party may first *choose* its own output, and then the honest party's output is computed with the reverse sampling algorithm for $\mathcal{C}$. As we show in the following, PCGs can be used to securely realize $\mathcal{F}^{\mathcal{C}}_{\mathsf{corr}*}$, opening up many important applications at no extra cost.

To realize $\mathcal{F}^{\mathcal{C}}_{\mathsf{corr}*}$, we use a simple protocol, $\Pi^{\mathcal{C}}_{\mathsf{corr}*}$, that calls $\mathcal{F}^{\mathsf{PCG.Gen}}_{\mathsf{corr}}$ so that each party obtains a seed k_σ, which is then expanded to get the output $\mathsf{PCG.Expand}(\sigma, \mathsf{k}_\sigma)$.

Theorem 66. *Let* $\mathsf{PCG} = (\mathsf{PCG.Gen}, \mathsf{PCG.Expand})$ *be a secure PCG for a reverse-sampleable correlation generator,* $\mathcal{C}$*. Then the protocol* $\Pi_{\mathsf{corr}*}$ *securely realizes the* $\mathcal{F}^{\mathcal{C}}_{\mathsf{corr}*}$ *functionality against a static, malicious adversary.*

Proof. Let Adv be a static adversary against the protocol π. We construct a simulator Sim, which interacts with Adv and $\mathcal{F}^{\mathcal{C}}_{\mathsf{corr}*}$ to produce a view for Adv that is indistinguishable from a real execution of the protocol. When both parties are corrupted, the simulator just runs Adv internally and security is straightforward. Similarly, when both parties are honest, simulation is trivial and indistinguishability follows from the correctness of PCG. Now suppose that only P_σ is corrupted, for $\sigma \in \{0,1\}$. On receiving the input 1^λ, Sim samples a pair of seeds $(\mathsf{k}_0, \mathsf{k}_1) \leftarrow_R \mathsf{PCG.Gen}(1^\lambda)$, then sends k_σ to Adv as its output of $\mathcal{F}^{\mathsf{PCG.Gen}}_{\mathsf{corr}}$, computes $\mathsf{R}_\sigma \leftarrow \mathsf{PCG.Expand}(\sigma, \mathsf{k}_\sigma)$ and sends this to $\mathcal{F}^{\mathcal{C}}_{\mathsf{corr}*}$. Notice that in the ideal execution, the view of the distinguisher consists of the seed k_σ and the honest party's output $\mathsf{R}_{1-\sigma}$, which is computed by $\mathcal{F}^{\mathcal{C}}_{\mathsf{corr}*}$ as $\mathsf{R}_{1-\sigma} \leftarrow_R \mathsf{RSample}(\sigma, \mathsf{R}_\sigma)$. The only difference in the real execution, is that there the honest party's output is computed with $\mathsf{PCG.Expand}(1-\sigma, \mathsf{k}_{1-\sigma})$. These two views are computationally indistinguishable, due to the security property of PCG. □

6.4 Our Generic Construction of a PCG

In this section we provide a high-level template for building a PCG. This can be viewed as a step forward towards towards the PCG construction given in Section 6.5.

On a high level, our strategy is as follows: We combine a pseudorandom generator for expanding a short seed into a long pseudorandom string with a homomorphic secret sharing scheme which allows to *locally* compute a correlation on given input shares.

More precisely, in this section we consider the special case, where R_0, R_1 are correlated if $\mathsf{R}_0 + \mathsf{R}_1 = f(X)$ for some input X and fixed function f. Now, consider a homomorphic secret sharing (HSS) scheme with additive reconstruction for f. Recall that given *shares of the input* X an HSS allows to locally evaluate f on the shares, s.t. the respective outputs add up to $f(X)$.

The idea for our generic PCG construction is as follows: During key generation a short seed k is shared between the players (as HSS shares). For expansion, the players can then locally evaluate $f(\mathsf{PRG}(k))$ via the HSS operations. By the correctness of the HSS that indeed gives outputs $\mathsf{R}_0, \mathsf{R}_1$ with $\mathsf{R}_0 + \mathsf{R}_1 = f(X)$, where $X = \mathsf{PRG}(k)$. In this section we formally prove that the described construction meets the PCG requirements.

Note that the challenge lies in actually instantiating the described approach efficiently. This is due to the fact that known efficient HSS constructions only apply to limited classes of functions, for instance Branching Programs. It is therefore crucial to carefully select the underlying PRG and HSS.

In the following we formally define additive correlations corresponding to a function, and give a generic construction of PCGs based on a pseudorandom generator and a suitable homomorphic secret scheme.

We will consider *additive correlations* corresponding to a family of functions $\mathcal{F}$. Such a correlation is generated by outputting an additive secret-sharing of a function from $f \in \mathcal{F}$ applied to a source of randomness.

Definition 67 (Correlation Generators for Additive Correlations)**.** Let R be a ring. Let $n, m \in \mathbb{N}$ and $\mathcal{F} \subseteq \{f \colon R^n \to R^m\}$ be a family of functions. Then we define a *correlation generator* $\mathcal{C}_\mathcal{F}$ *for* $\mathcal{F}$ as follows: On input 1^λ and $f \in \mathcal{F}$ the correlation generator $\mathcal{C}_\mathcal{F}$ samples $X \leftarrow_R R^n$, and returns a pair $(\mathsf{R}_0, \mathsf{R}_1) \in R^m \times R^m$, which is distributed uniformly at random conditioned on $\mathsf{R}_0 + \mathsf{R}_1 = f(X)$.

Note that $\mathcal{C}_\mathcal{F}$ is reverse-sampleable for any family of functions $\mathcal{F}$, as given a function $f \in \mathcal{F}$ and a share R_σ, one can draw an input $X \leftarrow_R R^n$ and set $\mathsf{R}_{1-\sigma} := \mathsf{R}_\sigma - f(X)$. Further, note that it is straightforward to includes shares of the inputs in the correlation by considering the family $\mathcal{F}' := \{f' \colon R^n \to R^{n+m}, X \mapsto (X, f(X)) \mid f \in \mathcal{F}\}$.

Definition 68 (HSS satisfying Pseudorandomness of Outputs)**.** We say an HSS $\mathsf{HSS} = (\mathsf{HSS.Gen}, \mathsf{HSS.Share}, \mathsf{HSS.Eval})$ for a function family $\mathcal{F} := \{f \colon R^n \to R^m\}$ satisfies *pseudorandomness of outputs*, if for all $f \colon R^n \to R^m \in \mathcal{F}, (\mathsf{sk}, \{\mathsf{ek}_\sigma\}_{\sigma \in \{0,1\}}) \leftarrow \mathsf{HSS.Gen}(1^\lambda)$, $X \leftarrow_R R^n$, $(\mathsf{k}_0, \mathsf{k}_1) \leftarrow_R \mathsf{Share}(\mathsf{sk}, X)$, and $\sigma \in \{0, 1\}$ the output $\mathsf{R}_\sigma \leftarrow_R \mathsf{HSS.Eval}(\sigma, \mathsf{ek}_\sigma, \mathsf{k}_\sigma, f)$ is distributed computationally close to uniformly at random over the output space.

- $\mathsf{PCG.Setup}(1^\lambda)$: Sample and output $(\mathsf{sk}, \{\mathsf{ek}_\sigma\}_{\sigma\in\{0,1\}}) \leftarrow \mathsf{HSS.Gen}(1^\lambda)$.
- $\mathsf{PCG.Gen}(\mathsf{sk})$: Sample $r \leftarrow R^\ell$ and output $(\mathsf{k}_0, \mathsf{k}_1) \leftarrow \mathsf{HSS.Share}(\mathsf{sk}, r)$.
- $\mathsf{PCG.Expand}(\sigma, \mathsf{ek}_\sigma, \mathsf{k}_\sigma, f)$: Output $\mathsf{R}_\sigma \leftarrow \mathsf{HSS.Eval}(\sigma, \mathsf{ek}_\sigma, \mathsf{k}_\sigma, f \circ \mathsf{PRG})$.

Figure 6.3: PCG for correlation $\mathcal{C}_\mathcal{F}$. Here, PRG is a PRG and HSS = (HSS.Gen, HSS.Share, HSS.Eval) an HSS for the family of functions $\mathcal{F}_{\mathsf{HSS}} := \{f \circ \mathsf{PRG}\colon r \mapsto f(\mathsf{PRG}(r)) \mid f \in \mathcal{F}\}$.

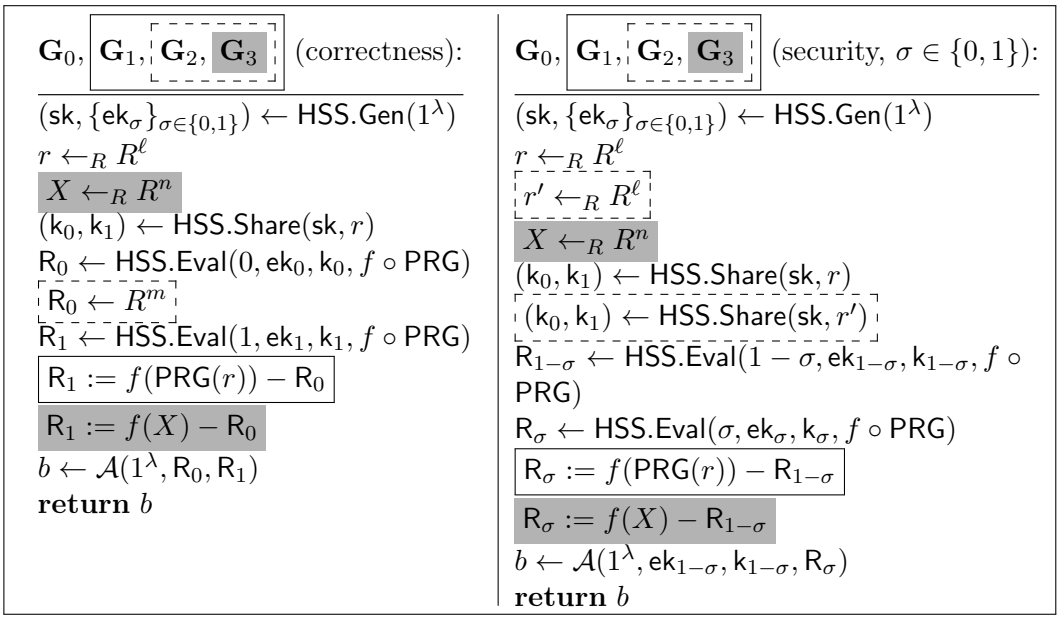

Figure 6.4: Games $\mathbf{G}_0$ and $\mathbf{G}_1$ in the proof of Theorem 69.

Note that if $f(\mathcal{U}(R^n))$ (where by $\mathcal{U}$ we denote the uniform distribution) is close to being uniformly random on R^m, this property follows from the security of HSS.

Theorem 69. *(PCG for Additive Correlations from HSS). Let R be a ring and $n, m, \ell \in \mathbb{N}$. Let $\mathcal{F} \subseteq \{f\colon R^n \to R^m\}$ be a family of functions. Let* PRG *be a PRG and* HSS = (HSS.Gen, HSS.Share, HSS.Eval) *a (secret-key) homomorphic secret sharing scheme with overhead O_{HSS}[2] for the family of functions $\mathcal{F}_{\mathsf{HSS}} := \{f \circ \mathsf{PRG}\colon R^n \to R^m, r \mapsto f(\mathsf{PRG}(r)) \mid f \in \mathcal{F}\}$ that further satisfies pseudorandomness of outputs. Then,* PCG = (PCG.Setup, PCG.Gen, PCG.Expand) *as defined in Figure 6.3 is a* PCG *for the correlation generator $\mathcal{C}_\mathcal{F}$ with key-length upper bounded by $\ell \cdot O_{\mathsf{HSS}}$.*

Proof. Correctness. Let $f \in \mathcal{F}$. We show correctnes via a series of games: The goal is to show computational indistinguishability between the output of the pseudorandom correlation generator (game $\mathbf{G}_0$) and the output of the correlation generator $\mathcal{C}_\mathcal{F}$ (game $\mathbf{G}_3$). By ε_i we denote the probablity that a PPT adversary $\mathcal{A}$ outputs 1 in game $\mathbb{G}_i$. For an overview of the games we refer to Figure 6.4.

[2]We say a HSS has *overhead* O_{HSS}, if for every input the share size does not exceed O_{HSS} times the input size.

Transition $\mathbf{G}_0 \rightsquigarrow \mathbf{G}_1$: By correctness of the HSS, we have

$$|\varepsilon_0 - \varepsilon_1| \leq 1 - \Pr^{\mathrm{cor}}_{\mathsf{HSS},r,\mathsf{PRG}\circ f}(\lambda) \leq \mathsf{negl}(\lambda)$$

for some negligible function $\mathsf{negl}\colon \mathbb{N} \to String^+$.

Transition $\mathbf{G}_1 \rightsquigarrow \mathbf{G}_2$: An adversary distinguishing between game $\mathbf{G}_1$ and $\mathbf{G}_2$ can be transformed into an adversary distinguishing the output R_0 from uniformly at random.

By pseudorandomness of the outputs of the HSS HSS, we thus have that

$$|\varepsilon_1 - \varepsilon_2|$$

is negligible in λ.

Transition $\mathbf{G}_2 \rightsquigarrow \mathbf{G}_3$: A distinguisher $\mathcal{A}$ between game $\mathbf{G}_2$ and game $\mathbf{G}_3$ can be transformed into an adversary $\mathcal{B}$ on the pseudorandomness of PRG as follows. The adversary $\mathcal{B}$ sets up everything according to $\mathbf{G}_3$, but sets $\mathsf{R}_1 := f(U) - \mathsf{R}_0$, where U is the output of $\mathcal{B}$'s pseudorandomness experiment. If U was pseudorandom, then $\mathcal{B}$ simulates $\mathbf{G}_2$, otherwise, $\mathcal{B}$ simulates $\mathbf{G}_3$. We thus have

$$|\varepsilon_2 - \varepsilon_3| \leq \mathrm{Adv}^{\mathrm{prg}}_{\mathsf{PRG},\mathcal{B}}(\lambda).$$

Security. Let $\sigma \in \{0,1\}$. We show security via a sequence of hybrid games, where in game $\mathbf{G}_0$ the adversary $\mathcal{A}$ obtains the output R_σ of the pseudorandom correlation generator, and in game $\mathbf{G}_3$ the adversary obtains a pseudorandom R_σ (conditioned on being related to $\mathsf{R}_{1-\sigma} \leftarrow \mathsf{PCG.Expand}(1-\sigma, \mathsf{k}_{1-\sigma})$). Again, by ε_i we denote the probablity that a PPT adversary $\mathcal{A}$ outputs 1 in game $\mathbb{G}_i$. For an overview of the games we refer to Figure 6.4. Transitions $\mathbf{G}_0 \rightsquigarrow \mathbf{G}_1$ and $\mathbf{G}_2 \rightsquigarrow \mathbf{G}_3$ are the same as in the proof of correctness. It is left to show transition $\mathbf{G}_1 \rightsquigarrow \mathbf{G}_2$:

Transition $\mathbf{G}_1 \rightsquigarrow \mathbf{G}_2$: We can transform an adversary $\mathcal{A}$ distinguishing between games $\mathbf{G}_1$ and $\mathbf{G}_2$ into an adversary $\mathcal{B}$ attacking the security of the homomorphic secret sharing scheme HSS as follows. The adversary $\mathcal{B}$ samples $r, r' \leftarrow R^\ell$. Next, $\mathcal{B}$ gives $b = 1 - \sigma$, r and r' to the HSS security experiment and receives an evaluation key ek_b and a share k_b. Now, $\mathcal{B}$ can compute $\mathsf{R}_{1-\sigma}$ by evaluating the HSS on ek_b and k_b and forward $\mathsf{ek}_b, \mathsf{k}_b$ and $\mathsf{R}_\sigma = f(\mathsf{PRG}(r)) - \mathsf{R}_{1-\sigma}$ to the adversary $\mathcal{A}$. If ek_b and k_b corresponds to an encryption of r, then $\mathcal{B}$ simulates $\mathbf{G}_1$, otherwise $\mathcal{B}$ simulates game $\mathbf{G}_2$. We thus have

$$|\varepsilon_1 - \varepsilon_2| \leq \mathrm{Adv}^{\mathrm{sec}}_{\mathsf{HSS},\mathcal{B}}(\lambda).$$

Altogether, we conclude that our generic PCG construction satisfies correctness and security. □

6.5 Our PCG for Authenticated Beaver Triples

In this section we give a lattice-based PCG construction for any family of polynomials of bounded degree over large finite fields. As a use case we consider the generation of authenticated Beaver triples, that is, the correlation

$$\{(a, b, ab, a\alpha, b\alpha, ab\alpha) \mid a, b \in \mathbb{Z}_r\}$$

for some fixed MAC key $\alpha \in \mathbb{Z}_r$.

A Pseudorandom Generator from MQ. Let R be a ring, and $\ell, n \in \mathbb{N}$. Let $\mathcal{M}(\ell^2, n, R)$ be a distribution over $R^{\ell^2 \times n}$ and $\mathbf{M} \leftarrow_R \mathcal{M}(\ell^2, n, R)$. We assume that for an appropriate choice of parameters

$$\mathsf{PRG}_{\mathsf{MQ}} \colon R^{\ell} \to R^{n}, \sigma \mapsto \mathbf{M}^{\top} \cdot (\sigma \otimes \sigma)$$

is a PRG. We say $\mathcal{M}(\ell^2, n, R)$ has *sparsity* ρ, if for every matrix $\mathbf{M}$ in the image of $\mathcal{M}(\ell^2, n, R)$, the number of non-zero entries in any column of $\mathbf{M}$ is at most ρ.

Note that if we choose $\mathcal{M}(\ell^2, n, R)$ to be the uniform distribution over $R^{\ell^2 \times n}$ the above assumption equals the MQ assumption of [MI88, Wol05, AHI+17]. While multivariate public-key cryptography has a long history of schemes being built then broken, we stress that the MQ assumption itself (which states that it is infeasible to solve a random system of quadratic equations) is believed to be a conservative assumption (in particular, the pseudorandomness of the MQ-based PRG reduces to the conjectured *one-wayness* of solving a random system of quadratic assumptions [BGP06]), and underlies the security of plausible and well-studied primitives in minicrypt (such as signatures scheme, or the stream cipher QUAD [BGP06]). Existing attacks on multivariate public-key cryptosystems all exploited the fact that the security of these systems did not in fact reduce to the MQ assumption. Furthermore, variants of MQ with a sparse matrix were considered several time as a natural optimization of MQ-based schemes [BCJ07, LLY08], and the resistance of the variant with sparse matrix against classical attacks was analyzed in [BCJ07, DyY07].

In the following we instantiate the generic construction with variants of RLWE-based homomorphic secret sharing schemes.

PCG from Somewhat Homomorphic Encryption. As observed in [DHRW16, BKS19], from a somewhat homomorphic encryption scheme which supports distributed decryption one can construct a homomorphic secret sharing scheme. In the following we give a semi-generic definition of properties the underlying encryption scheme has to satisfy. Note that the definition can be instantiated with a variety of lattice-based encryption schemes.

Definition 70 (Depth-d somewhat homomorphic encryption w/ distributed decryption)**.** Let $\mathsf{PKE} := (\mathsf{PKE.Gen}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ be an IND-CPA secure public-key encryption scheme. We say that PKE is a *secure depth-d public-key encryption scheme with distributed decryption* if it further satisfies the following properties:

- *Distributed decryption:* Let $R := \mathbb{Z}[X]/(X^N + 1)$, for N a power of two, $\kappa \in \mathbb{N}$ and the secret key space of PKE contained in R_q^{κ}. We say PKE supports distributed decryption, if there exists an algorithm DDec such that

for $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{PKE.Gen}(1^\lambda)$, $\mathsf{sk}_0 \leftarrow_R R_q^\kappa$, $\mathsf{sk}_1 := \mathsf{sk} - \mathsf{sk}_0$, $m \in R_r$, and $\mathbf{c} \leftarrow_R \mathsf{Enc}(\mathsf{pk}, m)$ it holds

$$\mathsf{DDec}(\mathsf{sk}_0, \mathbf{c}) + \mathsf{DDec}(\mathsf{sk}_1, \mathbf{c}) = m$$

with overwhelming probability.

- *Depth-d somewhat homomorphic encryption:* There exists a procedure $\mathsf{PKE.Eval}$ such that for any function $f\colon R^n \to R^m$ that can be evaluated by a circuit of depth at most d, for any $\lambda \in \mathbb{N}$, for any $(\mathsf{pk}, \mathsf{sk})$ in the image of $\mathsf{Gen}(1^\lambda)$, for all messages $m_1, \ldots, m_n \in R_r$, for all ciphertexts $\mathbf{c}_1, \ldots \mathbf{c}_n$ in the image of $\mathsf{PKE.Enc}(\mathsf{pk}, m_1), \ldots, \mathsf{PKE.Enc}(\mathsf{pk}, m_n)$ and for any $\mathbf{c}$ in the image of $\mathsf{PKE.Eval}(f, (\mathbf{c}_1, \ldots, \mathbf{c}_n))$ it holds

$$\mathsf{PKE.Dec}(\mathsf{sk}, \mathbf{c}) = f(m_1, \ldots, m_n).$$

Instantiating the generic construction with an HSS based on somewhat homomorphic encryption yields a PCG for any degree-d correlation (see Figure 6.5). As the following Theorem is a straightforward consequence of Theorem 69, we omit the proof.

Theorem 71. *Let R be a ring, $\ell, n, r, q, m \in \mathbb{N}$, PRG be a degree-c PRG $\mathsf{PRG}\colon R_r^\ell \to R_r^n$ and $\mathsf{PKE} = (\mathsf{PKE.Gen}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ be a depth-$\lceil \log cd \rceil$ somewhat homomorphic encryption scheme with message space R_r and secret key space contained in R_q^κ. If PKE additionally support distributed decryption, then the PCG $\mathsf{PCG} = (\mathsf{PCG.Setup}, \mathsf{PCG.Gen}, \mathsf{PCG.Expand})$ from Figure 6.5 is a PCG for the family of functions $\mathcal{F} := \{f\colon R_r^n \to R_r^m \mid f \text{ is of degree at most } d\}$.*

Remark 72. Note that the key generation of the PCG given in Figure 6.5 can be sourced out to the setup phase and the same secret key shares used across many instances.

Corollary 73. *Instantiating the PRG in the construction of Figure 6.5 with the degree-2 ρ-sparse PRG $\mathsf{PRG}_{\mathsf{MQ}} : \mathbb{Z}_r^\ell \to \mathbb{Z}_r^n$ and the somewhat homomorphic encryption scheme with the BGV encryption scheme of Brakerski et al. [BGV12] (chosing parameters $R = \mathbb{Z}[X]/(X^N + 1), r, q$ and error distribution χ s.t. evaluation of at least degree-5 functions/ depth-3 circuits is supported), we obtain a PCG for the generation of authenticated Beaver triples, assuming ρ-sparse $\mathcal{M}(\ell^2, n, R_r)$-MQ and $\mathsf{RLWE}_{N,q,\chi}$.*

Efficiency Estimates. Authenticated Beaver triples are used in multi-party protocols like [DPSZ12] to achieve fast online computation. Our PCG construction can be used as a plug-in to replace the preprocessing in such protocols by a short joint seed generation phase (with little communication) followed by a completely silent expansion phase. As in the described setting a large amount of Beaver triples has to be generated at once, lattice-based PCG constructions are of practical interest, despite the overhead introduced by encryption.

Generating many Beaver triples at once we can use ciphertext packing, as first observed by [SV14].

$\underline{\mathsf{PCG.Gen}(1^\lambda):}$

- **Generate the encryption keys.** Generate keys $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{PKE.Gen}(1^\lambda)$. Choose $\mathsf{sk}_0 \leftarrow_R R_q^\kappa$ and set $\mathsf{sk}_1 := \mathsf{sk} - \mathsf{sk}_0$.
- **Choose and encrypt a PRG-seed.** Choose $r \leftarrow \mathcal{D}^\ell(R_r)$. Compute

$$\mathbf{c}^r = \mathsf{PKE.Enc}(\mathsf{pk}, r) \in (R_q^\kappa)^\ell.$$

- Output $\mathsf{k}_0 := (\mathsf{sk}_0, \mathbf{c}^r)$, $\mathsf{k}_1 := (\mathsf{sk}_1, \mathbf{c}^r)$.

$\underline{\mathsf{PCG.Expand}(\sigma, \mathsf{k}_\sigma, f):}$

- Parse $\mathsf{k}_\sigma =: (\mathsf{sk}_\sigma, \mathbf{c}^r)$.
- **Evaluate $f \circ \mathsf{PRG}$ on the encrypted seed.** Compute

$$\mathbf{c}^Y \leftarrow \mathsf{PKE.Eval}(f \circ \mathsf{PRG}, \mathbf{c}^r) \in (R_q^\kappa)^m.$$

- **Decrypt the result.** Decrypt and output

$$\mathsf{R}_\sigma^Y \leftarrow \mathsf{PKE.DDec}(\mathsf{sk}_\sigma, \mathbf{c}^Y) \in R_r^m.$$

Figure 6.5: PCG for the family of degree-d functions from degree-c $\mathcal{D}^\ell$-PRG PRG and depth-$\lceil \log cd \rceil$ somewhat homomorphic encryption scheme PKE.

Remark 74 (Ciphertext packing, [SV14]). Let r be a prime and $N \in \mathbb{N}$ a power of 2, such that the polynomial $X^N + 1$ splits over $\mathbb{Z}_r$ into pairwise different degree-1 polynomials. If $R := \mathbb{Z}[X]/(X^N + 1)$ (similar for general cyclotomic polynomials), this implies $R_r \cong (\mathbb{Z}_r)^N$ and enables "packing" N plaintexts into one ciphertext (by encrypting $\psi(z)$ for some $z \in \mathbb{Z}_r^N$, where $\psi\colon (\mathbb{Z}_r)^N \to R_r$). In the following we will refer to R_r as *coefficient representation*, and to $(\mathbb{Z}_r)^N$ as *CRT representation*.

Thus, each ciphertext has room to hold N encryptions. We first consider "naive" ciphertext packing: We start with ℓ encryptions of each N seeds $r \in \mathbb{Z}_r^\ell$, perform the expansion homomorphically on the ciphertexts (which corresponds to expanding feach of the N seeds in parallel). This gives an output of nN correlated tuples in total.

In the following we estimate efficiency of the PCG construction given in Corollary 73 with the above described ciphertext packing. We use the parameters given in [CS16] to support depth-4 homomorphic operations (as an upper bound) and plaintext space modulus $\approx 2^{128}$ listed in the following. Here, by T_M we denote the time required for multiplication over R_q and by T_C the time for multiplication of a $\mathbb{Z}_q$ element with an element in R_q.

- *Dimension of R (over $\mathbb{Z}$):* $N \approx 13688$ (we use $N = 2^{14}$)
- *Ciphertext modulus:* $\log q \approx 750$ (we use $\log q = 744$)
- *Parameter for key switching:* $\log T \approx 140$

- *Cost of key switching:* $T_{\mathsf{KS}} \approx 2(\log q / \log T)T_C$
- *Cost of multiplication on ciphertexts:* $T_{\mathsf{Eval}} \approx 4T_M + T_{\mathsf{KS}}$
- *Cost of multiplication of constant with ciphertext:* $\approx 2T_C$
- *Cost of encryption:* $T_{\mathsf{Enc}} \approx 2(T_M + T_C)$
- *Cost of decryption:* $T_{\mathsf{Dec}} \approx 2T_M$

For MQ we use parameters $n = \ell^2/24$ and $\rho = 100$. Further, we set $\ell = c \cdot 2^9$ for $c \geq 1$. Later we will see that chosing $c = 1$ we surpass the breakeven point. In other words, $\ell = 2^9$ is the smallest choice where the total output size of the correlation generator exceeds the seed-length. Our runtime estimates are based on NFLLib [ABG+16]: A multiplication over R_q requires time ≈ 9.54 ms and a multiplication over $R_q \times \mathbb{Z}_q$ requires time ≈ 0.55 ms. For an overview of estimated setup computation and communication complexity (i.e. time and communication required for jointly generating the seed) and estimated expansion times for the described PCG construction and variants we refer to Table 6.1 in the introduction.

Distributed Seed Generation. We first describe the setup of the keys and MAC $\alpha \in \mathbb{Z}_r$, which can be reused across many instances. First, the parties jointly generate secret key shares $(\mathsf{sk}_0, \mathsf{sk}_1)$ and the corresponding public key pk, e.g. by generating secret keys according to a suitable distribution and exchanging shares as well as the corresponding public keys. Next, both parties choose a MAC share $\alpha_\sigma \leftarrow_R \mathbb{Z}_r$ and define $\alpha_\sigma \in \mathbb{Z}_r^N$ to be the vector of all α_σ entries. Next, the parties each compute and exchange $\mathbf{c}^{\psi(\alpha_\sigma)} := \mathsf{Enc}(\mathsf{pk}, \psi(\alpha_\sigma))$, and set $\mathbf{c}^{\psi(\alpha)} := \mathbf{c}^{\psi(\alpha_0)} + \mathbf{c}^{\psi(\alpha_1)}$.

To generate encryptions of N seeds a and b in $\mathbb{Z}_r^\ell$, both parties repeat 2ℓ times: Sample an element R_r at random (corresponding to N random $\mathbb{Z}_r$ elements), and as for generating an encryption of the MAC key, exchange and add up the corresponding encryption.

As computation and communication is dominated by the last step, a rough estimate in the semi-honest setting are as follows: Generating 2ℓ encryptions takes about $c \cdot 20$ seconds of computation and exchanging 2ℓ ciphertexts (each of size $2N \log q$ bits) requires $c \cdot 3$ GB of communication (per party). We estimate that in the dishonest setting communication complexity would roughly double.

Expansion Rate. We expand $2\ell N$ elements in $\mathbb{Z}_q$ to nN shared authenticated Beaver triples in $\mathbb{Z}_r$ (each consisting of 6 $\mathbb{Z}_r$ elements), which corresponds to expanding roughly $c \cdot 3$ GB of seed material to authenticated Beaver triples of total size $c^2 \cdot 17$ GB.

Computational Efficiency of Expansion. The computational costs add up as follows.

- *Expanding the seed:* The complexity to evaluate the PRG homomorphically on 2ℓ ciphertexts sums up to $2\ell^2$ ciphertext multiplications and $4n\rho$ multiplications of a constant with a ciphertext.
- *Computing the triples:* Evaluation of f_α requires $4n$ ciphertexted multiplications.

- *Obtaining the output shares:* To obtain the output we have to decrypt $n6$ ciphertexts.

Altogether, the costs sum up to

$$\approx 4n\rho T_C + 4(2\ell^2 + 7n)T_M + 2(\ell^2 + 2n)T_{\mathsf{KS}}.$$

This gives a total computation time of around $c^2 \cdot 8.0$ hours, which corresponds to an amortized computations time of roughly 0.16 ms per authenticated Beaver triple.

PCG from SHE with Nearly Linear Decryption In this section we consider a hybrid of the HSS based on somewhat homomorphic encryption and the HSS presented on the previous chapter based on encryption schemes which satisfy *nearly linear decryption*. Recall that the idea of our HSS is to replace multiplications of ciphertexts $\mathbf{c}^x$ and $\mathbf{c}^y$, by a distributed decryption of the ciphertext $\mathbf{c}^x$ with shares of $y \cdot \mathsf{sk}$.

Our strategy is to replace the last multiplication with an encryption of $\psi(\alpha)$ by a distributed decryption of $\psi(\alpha)$ times the secret key. This leads to an improvement in terms of computation, as in practise distributed decryption can be an order of magnitude faster than homomorphic multiplication.

In Figure 6.6 we present the construction for the generation of authenticated Beaver triples for a fixed MAC $\alpha \in \mathbb{Z}_r$ employing naive ciphertext packing.

When choosing the parameters of the underlying encryption scheme, one needs to take into account the noise growth introduced by homomorphic multiplication, as the distributed decryption technique of [BKS19] requires $r/\|e\|_\infty \ll q$, where e is the noise term in the ciphertext. We estimate that taking the parameters for depth-4 BGV scheme is sufficient in practice. With the scheme of Figure 6.6 we can save $3n$ evaluation operations compared to the scheme solely based on somewhat homomorphic encryption, which results in a saving of about $c^2 \cdot 0.4$ hours. We conjecture that an additional efficiency improvement over the previous scheme can be achieved by choosing the parameters more carefully.

We do not switch to the construction of Boyle et al. [BKS19] completely for the evaluation of f_α, even though this would save another n evaluation operations, as to evaluate polynomials of degree-3 or higher their construction relies on the so-called *modulus-lifting* technique, which necessitates the choice of larger parameters for the underlying ring to ensure correctness.

PCG with Iterative Expansion. As we choose a sparse matrix distribution (namely only $\rho = 100$ non-zero entries per row) to instantiate MQ, we can locally evaluate the PRG and therefore obtain a way to iteratively generate N Beaver triples at a time. This requires slightly more computational costs (assuming one wants to discard intermediary products), but allows to generate Beaver triples whenever needed instead of having to generate all at once. This can be achieved as follows: To generate shares of N Beaver triples, one needs to compute the scalar product of some i-th column of the $\mathcal{M}(l^2, n, R_r)$-MQ matrix $\mathbf{M}$ with the vector of encryptions of $r \otimes r$, where r is some seed. As $\mathbf{M}$ is sparse by assumption, this requires only to compute a linear combination of ρ products $r_i \cdot r_j$ on encryptions. With the numbers from above to generate N triples this approach inherits computational costs for expansion of

$$(4 + 2\rho)T_{\mathsf{Eval}} + 4\rho T_C + 12T_M = 4\rho T_C + (28 + 8\rho)T_M + (4 + 2\rho)T_{\mathsf{KS}}.$$

$\underline{\mathsf{PCG.Gen}(1^\lambda):}$

- **Generate the encryption keys.** Generate keys $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{PKE.Gen}(1^\lambda)$. Choose $\mathsf{sk}_0 \leftarrow_R R_q^\kappa$ and set $\mathsf{sk}_1 := \mathsf{sk} - \mathsf{sk}_0$.
- **Generate a share of the MAC key.** Choose $\alpha \leftarrow_R \mathbb{Z}_r$, define $\alpha \in \mathbb{Z}_r^N$ to be the vector of all α entries. Choose $\mathbf{s}_0 \leftarrow_R R_q^\kappa$ and compute
$$\mathbf{s}_1 := \psi(\alpha) \cdot \mathsf{sk} - \mathbf{s}_0.$$
- **Choose and encrypt a PRG-seeds.** Choose $r_a, r_b \leftarrow R_r^\ell$. Compute and output
$$\mathbf{c}^{r_a} = \mathsf{PKE.Enc}(\mathsf{pk}, r_a), \mathbf{c}^{r_b} = \mathsf{PKE.Enc}(\mathsf{pk}, r_b) \in (R_q^\kappa)^\ell.$$
- Output $\mathsf{k}_0 := (\mathsf{sk}_0, \mathbf{s}_0, \mathbf{c}^{r_a}, \mathbf{c}^{r_b})$, $\mathsf{k}_1 := (\mathsf{sk}_1, \mathbf{s}_1, \mathbf{c}^{r_a}, \mathbf{c}^{r_b})$.

$\underline{\mathsf{PCG.Expand}(\sigma, \mathsf{k}_\sigma, f):}$

- Parse $\mathsf{k}_\sigma =: (\mathsf{sk}_\sigma, \mathbf{s}_\sigma, \mathbf{c}^{r_a}, \mathbf{c}^{r_b})$.
- **Evaluate PRG homomorphically on the encrypted seed.** Compute
$$\mathbf{c}^a \leftarrow \mathsf{PKE.Eval}(\mathsf{PRG}, \mathbf{c}^{r_a}), \mathbf{c}^b \leftarrow \mathsf{PKE.Eval}(\mathsf{PRG}, \mathbf{c}^{r_b}) \in (R_q^\kappa)^n.$$
- **Evaluate F_α homomorphically on the encrypted input.** Compute
$$\mathbf{c}^Y \leftarrow \mathsf{PKE.Eval}(F_\alpha, \mathbf{c}^a, \mathbf{c}^b) \in (R_q^\kappa)^{6n}.$$
- **Obtain shares of Y via distributed decryption with sk_σ .** Compute
$$\mathsf{R}_\sigma^Y \leftarrow \mathsf{PKE.DDec}(\mathbf{s}_\sigma, \mathbf{c}^Y) \in R_r^{6n}.$$
- **Obtain the output shares over $\mathbb{Z}_r$.** Output
$$\Psi^{-1}(\mathsf{R}^Y) \in \mathbb{Z}_r^{6nN}.$$

Figure 6.6: PCG for authenticated Beaver triples with MAC α from degree-2 PRG $\mathsf{PRG}\colon \mathbb{Z}_r^\ell \to \mathbb{Z}_r^n$ and depth-2 somewhat homomorphic encryption scheme PKE with nearly linear decryption. Here, $F_\alpha\colon \mathbb{Z}_r^n \times \mathbb{Z}_r^n \to (\mathbb{Z}_r^n)^6, (a,b) \mapsto (a, b, a \circ b, a \circ \alpha, b \circ \alpha, a \circ b \circ \alpha)$ corresponds to evaluating f_α componentwise on each of the n input tuples ($\circ$ denotes the entrywise product). By Ψ^{-1} we denote the map evaluating $\psi^{-1}\colon R_r \to \mathbb{Z}_r^N$ componentwise.

This corresponds to a computation time of about 10 seconds per iteration (i.e. per N triples of total size 1.6 MB generated), which is amortized 0.57 seconds per triple. Note that this approach is still limited to a total expansion of nN triples.

PCG with Full Ciphertext Packing. Note that the above approach limits

us to go from $\approx \ell N$ elements to $\approx \ell^2 N$ elements (where N is the degree of the plaintext space over $\mathbb{Z}_r$). As N is generally quite large that leads to a somewhat limited expansion. To overcome this we investigated into packing more smartly, which allows going from $\approx N$ elements to $\approx N^2$ elements. To do so we build on the techniques of [HS18]. Even though the approach does not look promising in terms of computation due to expensive key switching operations, we believe that it is an interesting direction for future research.

For simplicity assume that $R := \mathbb{Z}[X]/(X^N+1)$ such that X^N+1 splits completely over $\mathbb{Z}_r$ and further, that the Galois group $G := \mathsf{Gal}(\mathbb{Q}(w)/\mathbb{Q}) = \{X \mapsto X^j : j \in \mathbb{Z}_m^\star\}$ is cyclic, where w is a primitive N-th root of unity. In this case the automorphisms $\alpha \in G$ act on $\mathbb{Z}_r^N$ by rotating the slots. Let $r = \psi(\mathbf{r}) \in R_r$ be some packed plain text in coefficient representation corresponding to a vector of N plain texts $\mathbf{r} \in \mathbb{Z}_r^N$. As shown in [LPR10, HS18], applying an automorphism τ to a ciphertext $\mathbf{c} \leftarrow_R \mathsf{Enc}(\mathsf{pk}, \psi(\mathbf{r}))$ leads to an encryption of $\psi(\tau(\mathbf{r}))$ which can be decrypted with $\tau(\mathsf{sk})$ (roughly, the reason is that applying an automorphism does not change the norm of the noise much and therefore one can decrypt with the shifted key to the shifted plaintext). Thus, by applying an automorphism and introducing a key-switching step, we can let plaintexts in different slots interact with each other and thereby achieve truly quadratic expansion.

Again, consider the BGV scheme with depth-4 homomorphic operations and plaintext space modulus $\approx 2^{128}$. (Note that in order to get exact numbers one would have to make a careful analysis on the the noise growth introduced by applying the automorphisms. As the numbers we use support up to depth-4 homomorphic operations, whereas we only need to compute a circuit of depth-3, we assume that the parameters also apply to the described setting for the following analysis.)

We apply the matrix multiplication technique of [HS14] and assume to be given $\mathbf{M} \leftarrow_R \mathcal{M}(N^2, N^2/24, R_q)$ accordingly in CRT-packed form. We can now start with a single ciphertext, i.e. $2N$ elements in $\mathbb{Z}_q$. In the following we provide the numbers for obtaining $\approx N^2$ authenticated Beaver triples. Additional costs are N key switching operations during seed generation, and N^2 key switching steps during matrix multiplication (because the matrix multiplication technique requires to permute each part of the vector). Note that one can alternatively only partly expand the ciphertext to generate less triples at a time and thereby saving in terms of computations (at a time). We obtain the following estimated costs:

$$\begin{aligned} &2NT_{\mathsf{KS}} + 2NT_{\mathsf{Eval}} + 2N^2T_{\mathsf{KS}} + 2(N^2/24)\rho + 4T_{\mathsf{Eval}} + 6T_{\mathsf{Dec}} \\ &\quad = (8N+28)T_M + 2(N^2+2N+2)T_{\mathsf{KS}} + 2(N^2/24)\rho. \end{aligned}$$

This adds up to a total expansion time in the order of $c^2 \cdot 900$ hours and is therefore far from practical. We leave it as an open question to achieve truly quadratic extension at a reasonable efficiency.

Bibliography

References

[ABG+16] Carlos Aguilar Melchor, Joris Barrier, Serge Guelton, Adrien Guinet, Marc-Olivier Killijian, and Tancrède Lepoint. NFLlib: NTT-based fast lattice library. In Kazue Sako, editor, *CT-RSA 2016*, volume 9610 of *LNCS*, pages 341–356. Springer, Heidelberg, February / March 2016.

[ACD+12] Masayuki Abe, Melissa Chase, Bernardo David, Markulf Kohlweiss, Ryo Nishimaki, and Miyako Ohkubo. Constant-size structure-preserving signatures: Generic constructions and simple assumptions. In Xiaoyun Wang and Kazue Sako, editors, *ASIACRYPT 2012*, volume 7658 of *LNCS*, pages 4–24. Springer, Heidelberg, December 2012.

[ACD+16] Masayuki Abe, Melissa Chase, Bernardo David, Markulf Kohlweiss, Ryo Nishimaki, and Miyako Ohkubo. Constant-size structure-preserving signatures: Generic constructions and simple assumptions. *Journal of Cryptology*, 29(4):833–878, October 2016.

[ACPS09] Benny Applebaum, David Cash, Chris Peikert, and Amit Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 595–618. Springer, Heidelberg, August 2009.

[ADK+13] Masayuki Abe, Bernardo David, Markulf Kohlweiss, Ryo Nishimaki, and Miyako Ohkubo. Tagged one-time signatures: Tight security and optimal tag size. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *PKC 2013*, volume 7778 of *LNCS*, pages 312–331. Springer, Heidelberg, February / March 2013.

[AHI+17] Benny Applebaum, Naama Haramaty, Yuval Ishai, Eyal Kushilevitz, and Vinod Vaikuntanathan. Low-complexity cryptographic hash functions. In Christos H. Papadimitriou, editor, *ITCS 2017*, volume 4266, pages 7:1–7:31, 67, January 2017. LIPIcs.

[AHN+17] Masayuki Abe, Dennis Hofheinz, Ryo Nishimaki, Miyako Ohkubo, and Jiaxin Pan. Compact structure-preserving signatures with almost tight security. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part II*, volume 10402 of *LNCS*, pages 548–580. Springer, Heidelberg, August 2017.

[AHY15] Nuttapong Attrapadung, Goichiro Hanaoka, and Shota Yamada. A framework for identity-based encryption with almost tight security. In Tetsu Iwata and Jung Hee Cheon, editors, *ASIACRYPT 2015, Part I*, volume 9452 of *LNCS*, pages 521–549. Springer, Heidelberg, November / December 2015.

[AJL+12] Gilad Asharov, Abhishek Jain, Adriana López-Alt, Eran Tromer, Vinod Vaikuntanathan, and Daniel Wichs. Multiparty computation with low communication, computation and interaction via threshold FHE. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 483–501. Springer, Heidelberg, April 2012.

[AJO+19] Masayuki Abe, Charanjit Jutla, Miyako Ohkubo, Jiaxin Pan, Arnab Roy, and Yuyu Wang. Shorter QA-NIZK and SPS with tighter security. In *ASIACRYPT 2019*. Springer, 2019.

[AJOR18] Masayuki Abe, Charanjit S. Jutla, Miyako Ohkubo, and Arnab Roy. Improved (almost) tightly-secure simulation-sound QA-NIZK with applications. In Thomas Peyrin and Steven Galbraith, editors, *ASIACRYPT 2018, Part I*, LNCS, pages 627–656. Springer, Heidelberg, December 2018.

[Ajt96] Miklós Ajtai. Generating hard instances of lattice problems (extended abstract). In *28th ACM STOC*, pages 99–108. ACM Press, May 1996.

[ASM07] Man Ho Au, Willy Susilo, and Yi Mu. Practical compact e-cash. In Josef Pieprzyk, Hossein Ghodosi, and Ed Dawson, editors, *ACISP 07*, volume 4586 of *LNCS*, pages 431–445. Springer, Heidelberg, July 2007.

[BBM00] Mihir Bellare, Alexandra Boldyreva, and Silvio Micali. Public-key encryption in a multi-user setting: Security proofs and improvements. In Bart Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 259–274. Springer, Heidelberg, May 2000.

[BCG+17] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, and Michele Orrù. Homomorphic secret sharing: Optimizations and applications. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 17*, pages 2105–2122. ACM Press, October / November 2017.

[BCG+19a] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, Peter Rindal, and Peter Scholl. Efficient two-round OT extension and silent non-interactive secure computation. In *ACM CCS 19*. ACM Press, 2019.

[BCG+19b] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl. Efficient pseudorandom correlation generators: Silent OT extension and more. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2019, Part III*, LNCS, pages 489–518. Springer, Heidelberg, August 2019.

[BCGI18] Elette Boyle, Geoffroy Couteau, Niv Gilboa, and Yuval Ishai. Compressing vector OLE. In *ACM CCS 18*, pages 896–912. ACM Press, 2018.

[BCJ07] Gregory V. Bard, Nicolas T. Courtois, and Chris Jefferson. Efficient methods for conversion and solution of sparse systems of low-degree multivariate polynomials over GF(2) via SAT-Solvers. Cryptology ePrint Archive, Report 2007/024, 2007. `http://eprint.iacr.org/2007/024`.

[BCKL08] Mira Belenkiy, Melissa Chase, Markulf Kohlweiss, and Anna Lysyanskaya. P-signatures and noninteractive anonymous credentials. In Ran Canetti, editor, *TCC 2008*, volume 4948 of *LNCS*, pages 356–374. Springer, Heidelberg, March 2008.

[BCKL09] Mira Belenkiy, Melissa Chase, Markulf Kohlweiss, and Anna Lysyanskaya. Compact e-cash and simulatable VRFs revisited. In Hovav Shacham and Brent Waters, editors, *PAIRING 2009*, volume 5671 of *LNCS*, pages 114–131. Springer, Heidelberg, August 2009.

[BDH14] Florian Böhl, Gareth T. Davies, and Dennis Hofheinz. Encryption schemes secure under related-key and key-dependent message attacks. In Hugo Krawczyk, editor, *PKC 2014*, volume 8383 of *LNCS*, pages 483–500. Springer, Heidelberg, March 2014.

[BDIR18] Fabrice Benhamouda, Akshay Degwekar, Yuval Ishai, and Tal Rabin. On the local leakage resilience of linear secret sharing schemes. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part I*, volume 10991 of *LNCS*, pages 531–561. Springer, Heidelberg, August 2018.

[BDOZ11] Rikke Bendlin, Ivan Damgård, Claudio Orlandi, and Sarah Zakarias. Semi-homomorphic encryption and multiparty computation. In Kenneth G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 169–188. Springer, Heidelberg, May 2011.

[Bea92] Donald Beaver. Efficient multiparty protocols using circuit randomization. In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 420–432. Springer, Heidelberg, August 1992.

[Bea96] Donald Beaver. Correlated pseudorandomness and the complexity of private computations. In *28th ACM STOC*, pages 479–488. ACM Press, May 1996.

[BFM88] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications (extended abstract). In *20th ACM STOC*, pages 103–112. ACM Press, May 1988.

[BG90] Mihir Bellare and Shafi Goldwasser. New paradigms for digital signatures and message authentication based on non-interative zero knowledge proofs. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 194–211. Springer, Heidelberg, August 1990.

[BGI15] Elette Boyle, Niv Gilboa, and Yuval Ishai. Function secret sharing. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 337–367. Springer, Heidelberg, April 2015.

[BGI16a] Elette Boyle, Niv Gilboa, and Yuval Ishai. Breaking the circuit size barrier for secure computation under DDH. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part I*, volume 9814 of *LNCS*, pages 509–539. Springer, Heidelberg, August 2016.

[BGI16b] Elette Boyle, Niv Gilboa, and Yuval Ishai. Function secret sharing: Improvements and extensions. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 16*, pages 1292–1303. ACM Press, October 2016.

[BGI+18] Elette Boyle, Niv Gilboa, Yuval Ishai, Huijia Lin, and Stefano Tessaro. Foundations of homomorphic secret sharing. In Anna R. Karlin, editor, *ITCS 2018*, volume 94, pages 21:1–21:21. LIPIcs, January 2018.

[BGP06] Côme Berbain, Henri Gilbert, and Jacques Patarin. QUAD: A practical stream cipher with provable security. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 109–128. Springer, Heidelberg, May / June 2006.

[BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In Shafi Goldwasser, editor, *ITCS 2012*, pages 309–325. ACM, January 2012.

[BHHO08] Dan Boneh, Shai Halevi, Michael Hamburg, and Rafail Ostrovsky. Circular-secure encryption from decision Diffie-Hellman. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 108–125. Springer, Heidelberg, August 2008.

[BJLS16] Christoph Bader, Tibor Jager, Yong Li, and Sven Schäge. On the impossibility of tight cryptographic reductions. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 273–304. Springer, Heidelberg, May 2016.

[BKP14] Olivier Blazy, Eike Kiltz, and Jiaxin Pan. (Hierarchical) identity-based encryption from affine message authentication. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 408–425. Springer, Heidelberg, August 2014.

[BKS19] Elette Boyle, Lisa Kohl, and Peter Scholl. Homomorphic secret sharing from lattices without FHE. In Vincent Rijmen and Yuval Ishai, editors, *EUROCRYPT 2019, Part II*, LNCS, pages 3–33. Springer, Heidelberg, May 2019.

[BLP+13] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. Classical hardness of learning with errors. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 575–584. ACM Press, June 2013.

[BPR12] Abhishek Banerjee, Chris Peikert, and Alon Rosen. Pseudorandom functions and lattices. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 719–737. Springer, Heidelberg, April 2012.

[Bra12] Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical GapSVP. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 868–886. Springer, Heidelberg, August 2012.

[BV11] Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from ring-LWE and security for key dependent messages. In Phillip Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 505–524. Springer, Heidelberg, August 2011.

[CDM+18] Geoffroy Couteau, Aurélien Dupin, Pierrick Méaux, Mélissa Rossi, and Yann Rotella. On the concrete security of Goldreich's pseudorandom generator. In Thomas Peyrin and Steven Galbraith, editors, *ASIACRYPT 2018, Part II*, LNCS, pages 96–124. Springer, Heidelberg, December 2018.

[CGKS95] Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan. Private information retrieval. In *36th FOCS*, pages 41–50. IEEE Computer Society Press, October 1995.

[Cle91] Richard Cleve. Towards optimal simulations of formulas by bounded-width programs. *Computational Complexity*, 1:91–105, 1991.

[Cor02] Jean-Sébastien Coron. Optimal security proofs for PSS and other signature schemes. In Lars R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 272–287. Springer, Heidelberg, April / May 2002.

[CS98] Ronald Cramer and Victor Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In Hugo Krawczyk, editor, *CRYPTO'98*, volume 1462 of *LNCS*, pages 13–25. Springer, Heidelberg, August 1998.

[CS03] Ronald Cramer and Victor Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM Journal on Computing*, 33(1):167–226, 2003.

[CS16] Ana Costache and Nigel P. Smart. Which ring based somewhat homomorphic encryption scheme is best? In Kazue Sako, editor, *CT-RSA 2016*, volume 9610 of *LNCS*, pages 325–340. Springer, Heidelberg, February / March 2016.

[CW13] Jie Chen and Hoeteck Wee. Fully, (almost) tightly secure IBE and dual system groups. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 435–460. Springer, Heidelberg, August 2013.

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[DHRW16] Yevgeniy Dodis, Shai Halevi, Ron D. Rothblum, and Daniel Wichs. Spooky encryption and its applications. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part III*, volume 9816 of *LNCS*, pages 93–122. Springer, Heidelberg, August 2016.

[DKPW12] Yevgeniy Dodis, Eike Kiltz, Krzysztof Pietrzak, and Daniel Wichs. Message authentication, revisited. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 355–374. Springer, Heidelberg, April 2012.

[DPSZ12] Ivan Damgård, Valerio Pastro, Nigel P. Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 643–662. Springer, Heidelberg, August 2012.

[DyY07] Jintai Ding and Bo yin Yang. Multivariates polynomials for hashing. Cryptology ePrint Archive, Report 2007/137, 2007. `http://eprint.iacr.org/2007/137`.

[DZ13] Ivan Damgård and Sarah Zakarias. Constant-overhead secure computation of Boolean circuits using preprocessing. In Amit Sahai, editor, *TCC 2013*, volume 7785 of *LNCS*, pages 621–641. Springer, Heidelberg, March 2013.

[EGL85] Shimon Even, Oded Goldreich, and Abraham Lempel. How to exchange secrets with oblivious transfer. In *Communications of the ACM, 28(6)*, pages 637–647. Aiken Computation Lab, Harvard University, 1985.

[EHK+13] Alex Escala, Gottfried Herold, Eike Kiltz, Carla Ràfols, and Jorge Villar. An algebraic framework for Diffie-Hellman assumptions. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 129–147. Springer, Heidelberg, August 2013.

[FV12] Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Report 2012/144, 2012. `http://eprint.iacr.org/2012/144`.

[GCD+16] Junqing Gong, Jie Chen, Xiaolei Dong, Zhenfu Cao, and Shaohua Tang. Extended nested dual system groups, revisited. In Chen-Mou Cheng, Kai-Min Chung, Giuseppe Persiano, and Bo-Yin Yang, editors, *PKC 2016, Part I*, volume 9614 of *LNCS*, pages 133–163. Springer, Heidelberg, March 2016.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *41st ACM STOC*, pages 169–178. ACM Press, May / June 2009.

[GHK17] Romain Gay, Dennis Hofheinz, and Lisa Kohl. Kurosawa-desmedt meets tight security. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part III*, volume 10403 of *LNCS*, pages 133–160. Springer, Heidelberg, August 2017.

[GHKP18] Romain Gay, Dennis Hofheinz, Lisa Kohl, and Jiaxin Pan. More efficient (almost) tightly secure structure-preserving signatures. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 230–258. Springer, Heidelberg, April / May 2018.

[GHKW16] Romain Gay, Dennis Hofheinz, Eike Kiltz, and Hoeteck Wee. Tightly CCA-secure encryption without pairings. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part I*, volume 9665 of *LNCS*, pages 1–27. Springer, Heidelberg, May 2016.

[GI99] Niv Gilboa and Yuval Ishai. Compressing cryptographic resources. In Michael J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 591–608. Springer, Heidelberg, August 1999.

[GI14] Niv Gilboa and Yuval Ishai. Distributed point functions and their applications. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 640–658. Springer, Heidelberg, May 2014.

[GL07] Jens Groth and Steve Lu. A non-interactive shuffle with pairing based verifiability. In Kaoru Kurosawa, editor, *ASIACRYPT 2007*, volume 4833 of *LNCS*, pages 51–67. Springer, Heidelberg, December 2007.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred Aho, editor, *19th ACM STOC*, pages 218–229. ACM Press, May 1987.

[Gol00] Oded Goldreich. Candidate one-way functions based on expander graphs. Cryptology ePrint Archive, Report 2000/063, 2000. `http://eprint.iacr.org/2000/063`.

[GOS12] Jens Groth, Rafail Ostrovsky, and Amit Sahai. New techniques for noninteractive zero-knowledge. *J. ACM*, 59(3):11:1–11:35, June 2012.

[GS08] Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 415–432. Springer, Heidelberg, April 2008.

[HHK18] Julia Hesse, Dennis Hofheinz, and Lisa Kohl. On tightly secure non-interactive key exchange. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 65–94. Springer, Heidelberg, August 2018.

[HIJ+16] Shai Halevi, Yuval Ishai, Abhishek Jain, Eyal Kushilevitz, and Tal Rabin. Secure multiparty computation with general interaction patterns.

In Madhu Sudan, editor, *ITCS 2016*, pages 157–168. ACM, January 2016.

[HJ12] Dennis Hofheinz and Tibor Jager. Tightly secure signatures and public-key encryption. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 590–607. Springer, Heidelberg, August 2012.

[HJ16] Dennis Hofheinz and Tibor Jager. Verifiable random functions from standard assumptions. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A, Part I*, volume 9562 of *LNCS*, pages 336–362. Springer, Heidelberg, January 2016.

[HK07] Dennis Hofheinz and Eike Kiltz. Secure hybrid encryption from weakened key encapsulation. In Alfred Menezes, editor, *CRYPTO 2007*, volume 4622 of *LNCS*, pages 553–571. Springer, Heidelberg, August 2007.

[HKS15] Dennis Hofheinz, Jessica Koch, and Christoph Striecks. Identity-based encryption with (almost) tight security in the multi-instance, multi-ciphertext setting. In Jonathan Katz, editor, *PKC 2015*, volume 9020 of *LNCS*, pages 799–822. Springer, Heidelberg, March / April 2015.

[HLLG19] Shuai Han, Shengli Liu, Lin Lyu, and Dawu Gu. Tight leakage-resilient CCA-security from quasi-adaptive hash proof system. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2019, Part II*, LNCS, pages 417–447. Springer, Heidelberg, August 2019.

[HLR07] Chun-Yuan Hsiao, Chi-Jen Lu, and Leonid Reyzin. Conditional computational entropy, or toward separating pseudoentropy from compressibility. In Moni Naor, editor, *EUROCRYPT 2007*, volume 4515 of *LNCS*, pages 169–186. Springer, Heidelberg, May 2007.

[Hof12] Dennis Hofheinz. All-but-many lossy trapdoor functions. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 209–227. Springer, Heidelberg, April 2012.

[Hof16] Dennis Hofheinz. Algebraic partitioning: Fully compact and (almost) tightly secure cryptography. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A, Part I*, volume 9562 of *LNCS*, pages 251–281. Springer, Heidelberg, January 2016.

[Hof17] Dennis Hofheinz. Adaptive partitioning. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part III*, volume 10212 of *LNCS*, pages 489–518. Springer, Heidelberg, April / May 2017.

[HPS18] Shai Halevi, Yuriy Polyakov, and Victor Shoup. An improved RNS variant of the BFV homomorphic encryption scheme. Cryptology ePrint Archive, Report 2018/117, 2018. `https://eprint.iacr.org/2018/117`.

[HS14] Shai Halevi and Victor Shoup. Algorithms in HElib. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 554–571. Springer, Heidelberg, August 2014.

[HS18] Shai Halevi and Victor Shoup. Faster homomorphic linear transformations in HElib. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part I*, volume 10991 of *LNCS*, pages 93–120. Springer, Heidelberg, August 2018.

[HW15] Pavel Hubacek and Daniel Wichs. On the communication complexity of secure function evaluation with long output. In Tim Roughgarden, editor, *ITCS 2015*, pages 163–172. ACM, January 2015.

[IKNP03] Yuval Ishai, Joe Kilian, Kobbi Nissim, and Erez Petrank. Extending oblivious transfers efficiently. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 145–161. Springer, Heidelberg, August 2003.

[ILL89] Russell Impagliazzo, Leonid A. Levin, and Michael Luby. Pseudo-random generation from one-way functions (extended abstracts). In *21st ACM STOC*, pages 12–24. ACM Press, May 1989.

[JOR18] Charanjit S. Jutla, Miyako Ohkubo, and Arnab Roy. Improved (almost) tightly-secure structure-preserving signatures. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018, Part II*, volume 10770 of *LNCS*, pages 123–152. Springer, Heidelberg, March 2018.

[JR14] Charanjit S. Jutla and Arnab Roy. Switching lemma for bilinear tests and constant-size NIZK proofs for linear subspaces. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part II*, volume 8617 of *LNCS*, pages 295–312. Springer, Heidelberg, August 2014.

[JR17] Charanjit S. Jutla and Arnab Roy. Improved structure preserving signatures under standard bilinear assumptions. In Serge Fehr, editor, *PKC 2017, Part II*, volume 10175 of *LNCS*, pages 183–209. Springer, Heidelberg, March 2017.

[KD04] Kaoru Kurosawa and Yvo Desmedt. A new paradigm of hybrid encryption scheme. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 426–442. Springer, Heidelberg, August 2004.

[Kil88] Joe Kilian. Founding cryptography on oblivious transfer. In *20th ACM STOC*, pages 20–31. ACM Press, May 1988.

[Koh19] Lisa Kohl. Hunting and gathering - verifiable random functions from standard assumptions with short proofs. In *PKC 2019, Part II*, LNCS, pages 408–437. Springer, Heidelberg, 2019.

[KPG99] Aviad Kipnis, Jacques Patarin, and Louis Goubin. Unbalanced oil and vinegar signature schemes. In Jacques Stern, editor, *EUROCRYPT'99*, volume 1592 of *LNCS*, pages 206–222. Springer, Heidelberg, May 1999.

[KPR18] Marcel Keller, Valerio Pastro, and Dragos Rotaru. Overdrive: Making SPDZ great again. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part III*, volume 10822 of *LNCS*, pages 158–189. Springer, Heidelberg, April / May 2018.

[KPW15] Eike Kiltz, Jiaxin Pan, and Hoeteck Wee. Structure-preserving signatures from standard assumptions, revisited. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 275–295. Springer, Heidelberg, August 2015.

[KW15] Eike Kiltz and Hoeteck Wee. Quasi-adaptive NIZK for linear subspaces revisited. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 101–128. Springer, Heidelberg, April 2015.

[LJYP14] Benoît Libert, Marc Joye, Moti Yung, and Thomas Peters. Concise multi-challenge CCA-secure encryption and signatures with almost tight security. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part II*, volume 8874 of *LNCS*, pages 1–21. Springer, Heidelberg, December 2014.

[LLHG18] Lin Lyu, Shengli Liu, Shuai Han, and Dawu Gu. Tightly SIM-SO-CCA secure public key encryption from standard assumptions. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018, Part I*, volume 10769 of *LNCS*, pages 62–92. Springer, Heidelberg, March 2018.

[LLY08] Feng-Hao Liu, Chi-Jen Lu, and Bo-Yin Yang. Secure PRNGs from specialized polynomial maps over any. In Johannes Buchmann and Jintai Ding, editors, *Post-quantum cryptography, second international workshop, PQCRYPTO 2008*, pages 181–202. Springer, Heidelberg, October 2008.

[LPJY15] Benoît Libert, Thomas Peters, Marc Joye, and Moti Yung. Compactly hiding linear spans - tightly secure constant-size simulation-sound QA-NIZK proofs and applications. In Tetsu Iwata and Jung Hee Cheon, editors, *ASIACRYPT 2015, Part I*, volume 9452 of *LNCS*, pages 681–707. Springer, Heidelberg, November / December 2015.

[LPR10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 1–23. Springer, Heidelberg, May / June 2010.

[LPR13] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. A toolkit for ring-LWE cryptography. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 35–54. Springer, Heidelberg, May 2013.

[LPY15] Benoît Libert, Thomas Peters, and Moti Yung. Short group signatures via structure-preserving signatures: Standard model security from simple assumptions. In Rosario Gennaro and Matthew J. B. Robshaw,

editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 296–316. Springer, Heidelberg, August 2015.

[LS15] Adeline Langlois and Damien Stehlé. Worst-case to average-case reductions for module lattices. *Des. Codes Cryptography*, 75(3):565–599, 2015.

[MI88] Tsutomu Matsumoto and Hideki Imai. Public quadratic polynominal-tuples for efficient signature-verification and message-encryption. In C. G. Günther, editor, *EUROCRYPT'88*, volume 330 of *LNCS*, pages 419–453. Springer, Heidelberg, May 1988.

[MR02] Silvio Micali and Ronald L. Rivest. Micropayments revisited. In Bart Preneel, editor, *CT-RSA 2002*, volume 2271 of *LNCS*, pages 149–163. Springer, Heidelberg, February 2002.

[MRV99] Silvio Micali, Michael O. Rabin, and Salil P. Vadhan. Verifiable random functions. In *40th FOCS*, pages 120–130. IEEE Computer Society Press, October 1999.

[MRV16] Paz Morillo, Carla Ràfols, and Jorge Luis Villar. The kernel matrix Diffie-Hellman assumption. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part I*, volume 10031 of *LNCS*, pages 729–758. Springer, Heidelberg, December 2016.

[MST03] Elchanan Mossel, Amir Shpilka, and Luca Trevisan. On e-biased generators in NC0. In *44th FOCS*, pages 136–145. IEEE Computer Society Press, October 2003.

[NNOB12] Jesper Buus Nielsen, Peter Sebastian Nordholt, Claudio Orlandi, and Sai Sheshank Burra. A new approach to practical active-secure two-party computation. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 681–700. Springer, Heidelberg, August 2012.

[NY90] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *22nd ACM STOC*, pages 427–437. ACM Press, May 1990.

[Pei09] Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In Michael Mitzenmacher, editor, *41st ACM STOC*, pages 333–342. ACM Press, May / June 2009.

[Rab81] Michael O. Rabin. How to exchange secrets with oblivious transfer. In *TR-81 edition*. Aiken Computation Lab, Harvard University, 1981.

[Ràf15] Carla Ràfols. Stretching groth-sahai: NIZK proofs of partial satisfiability. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 247–276. Springer, Heidelberg, March 2015.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th ACM STOC*, pages 84–93. ACM Press, May 2005.

[Sch18] Peter Scholl. Extending oblivious transfer with low communication via key-homomorphic PRFs. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018, Part I*, volume 10769 of *LNCS*, pages 554–583. Springer, Heidelberg, March 2018.

[SV11] N.P. Smart and F. Vercauteren. Fully homomorphic SIMD operations. Cryptology ePrint Archive, Report 2011/133, 2011. `http://eprint.iacr.org/2011/133`.

[SV14] N. P. Smart and F. Vercauteren. Fully homomorphic simd operations. *Des. Codes Cryptography*, 71(1):57–81, April 2014.

[Wol05] Christopher Wolf. Multivariate quadratic polynomials in public key cryptography. Cryptology ePrint Archive, Report 2005/393, 2005. `http://eprint.iacr.org/2005/393`.

[WRK17a] Xiao Wang, Samuel Ranellucci, and Jonathan Katz. Authenticated garbling and efficient maliciously secure two-party computation. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 17*, pages 21–37. ACM Press, October / November 2017.

[WRK17b] Xiao Wang, Samuel Ranellucci, and Jonathan Katz. Global-scale secure multiparty computation. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 17*, pages 39–56. ACM Press, October / November 2017.

[WYG+17] Frank Wang, Catherine Yun, Shafi Goldwasser, Vinod Vaikuntanathan, and Matei Zaharia. Splinter: Practical private queries on public data. In *14th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2017*, pages 299–313, 2017.

[Yao82a] Andrew Chi-Chih Yao. Protocols for secure computations (extended abstract). In *23rd FOCS*, pages 160–164. IEEE Computer Society Press, November 1982.

[Yao82b] Andrew Chi-Chih Yao. Theory and applications of trapdoor functions (extended abstract). In *23rd FOCS*, pages 80–91. IEEE Computer Society Press, November 1982.

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th FOCS*, pages 162–167. IEEE Computer Society Press, October 1986.