Karlsruhe Institute of Technology

Multilinear Maps in Cryptography

zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften / Ingenieurwissenschaften

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

genehmigte

Dissertation

von

Julia Hesse

aus Seeheim-Jugenheim

Tag der mündlichen Prüfung: 02.11.2016

Erster Gutachter: Prof. Dr. Dennis Hofheinz
Zweiter Gutachter: Prof. Dr. Marc Fischlin

Für

Elin

und

Niki.

♡

Danksagung

Kryptographie! Wer hätte es gedacht? Meine Eltern hätten vermutlich eher auf ein Gebiet gewettet, das mit technischen Zeichnungen einhergeht. Nun, dann hätten sie mir damals nicht dieses Buch über Geheimschriften geben sollen. Dies war aber auch ihr letzter Fehler, denn ab hier haben andere übernommen. Allen voran möchte ich Willi Geiselmann und Jörn Müller-Quade nennen, die seit vielen Jahren Vorlesungen über Kryptographie in Karlsruhe halten, die nicht nur mir Lust auf mehr gemacht haben. Für Jörns anschließendes Vertrauen bin ich bis heute sehr dankbar, denn dies zusammen mit seiner herzlichen Art hat mich damals endgültig überzeugt, den Schritt zur Promotion zu wagen.

Dennis, der diese Arbeit inhaltlich betreut hat, habe ich meine Ausbildung sowohl in kryptographischer als auch in wissenschaftlicher Hinsicht zu verdanken. Seine Passion für ungelöste theoretische Probleme hatte und hat auf mich eine äußerst ansteckende Wirkung. Wer Dennis kennt, der weiß, dass er mit Hingabe alles und jeden analysiert. Kryptographie, Politik, die letzte Doppelkopfrunde und - mit einer gehörigen Portion Ehrlichkeit und Selbstironie - auch uns selbst. Was hatten wir Spaß! Danke, Dennis, für diese schöne Zeit.

Was wäre die Arbeit ohne nette Kollegen, Mitautoren, Freunde...? Viele von Ihnen haben meine Zeit am Institut ungemein bereichert. Allen voran meine lieben Freunde Andy und Flo, deren Motivation und emotionaler Beistand auch erheblich zur Fertigstellung dieser Arbeit beigetragen hat. Ich danke Jessi für viele lustige Momente (besonders in Hotelzimmern). Dirk, der immer eine passende Katze oder Würschtl gefunden hat. Lisa, die ich am liebsten mitgenommen hätte. Der Doppelkopfgruppe und dem Kickerteam für lustige Abende und hochemotionale Mittagspausen-Fights. Brandon für seine Oma (die den besten Kartoffelsalat der Welt macht!). Bernhard für viele Gespräche und leckeren Obstsalat. Frau Manietta für die vielen Nerven, wenn ich mal wieder die Elternzeit falsch beantragt hatte. Marc für den netten Empfang und das Begutachten dieser Dissertation. Kenny for inviting me to RHUL (twice!). Eduarda for being the most annoying and most wonderful co-author I had so far. Alex, Anna-Louise, Antonio, Björn, Carmen, Christoph, Daniel, David, Erik, Gunnar, Holger, Jiaxin, Jochen, Mario, Matthias N, Matthias G, Matthias H, Patrik, Rafael, Simon, Tibor, Tobi, Willi, es war schön mit euch!

Meine Eltern haben mich mein ganzes Leben lang unterstützt und mir zur Seite gestanden. Egal ob Blumenladen oder Mathestudium. Es erscheint mir absurd, dies hier zu Papier zu bringen und ich hoffe, dass ich meine Dankbarkeit dafür noch sehr, sehr viele Male anders ausdrücken kann.

Zu guter Letzt danke ich meiner kleinen Familie, die mit Abstand am meisten unter der Fertigstellung dieser Arbeit gelitten hat. Ohne Nikis Unterstützung und sein Verständnis wäre das alles nicht möglich gewesen.

Contents

Zusammenfassung

WAS SIND MULTILINEARE ABBILDUNGEN? Multilineare Abbildungen sind mächtige Primitive in der Kryptographie. Beschrieben sind sie schnell: sie bilden k Gruppen ab in eine weitere Gruppe und sind linear in jeder Komponente. Notiert man die Gruppen (auch *multilineare Gruppen genannt*) multiplikativ und nennt die Abbildung e, dann gibt uns die Linearität folgende Gleichung:

$$e(g_1^{a_1}, \ldots, g_k^{a_k}) = g_T^{\prod_{i=1}^{k} a_i}.$$

Aus dieser Gleichung wird ersichtlich, dass uns e ermöglicht, ohne Kenntnis der Exponenten Produkte "im Exponenten" zu bilden (auch wenn wir dabei in einer neuen Gruppe landen).

MULTILINEARE ABBILDUNGEN IN DER KRYPTOGRAPHIE. In der Kryptographie ist man hauptsächlich interessiert an multilinearen Gruppen, in denen schwere Berechnungsprobleme existieren. Dabei gilt ein Problem als schwer, wenn kein effizienter Lösungsalgorithmus bekannt ist. Ein Beispiel für ein solches, im Zusammenhang mit multilinearen Abbildungen besonders interessantes Problem ist die Berechnung des diskreten Logarithmus (DLOG). Dessen Schwierigkeit ist die Voraussetzung dafür, dass Multiplikation im Exponenten nicht "einfach so", also ohne Verwendung der multilinearen Abbildung möglich ist. Tatsächlich basiert die Sicherheit von kryptographischen Konstruktionen in multilinearen Gruppen jedoch auf schwierigeren Problemen, die mit Hilfe der multilinearen Abbildung nicht trivial lösbar sind. Beispiele für solche Konstruktionen sind funktionale Verschlüsselung (*multi-input functional encryption*), nicht interaktiver Schlüsselaustausch zwischen vielen Benutzern (*non-interactive multi-user key exchange*) und ununterscheidbare Programmobfuszierung (*indistinguishability obfuscation*).

EIGENER BEITRAG. Das Forschungsgebiet der multilinearen Abbildungen und ihrer Verwendung in der Kryptographie ist noch recht jung. Beispielsweise wurden Kandidaten für k-lineare Abbildungen für $k > 2$ erst vor wenigen Jahren vorgeschlagen. In dieser Dissertation leisten wir die folgenden Beiträge zu diesem Gebiet:

- Existenz: Wir geben einen neuen Kandidaten für eine (Approximation einer) multilinearen Abbildung an. Unser Kandidat bietet zusätzliche Funktionalität indem er paarweises Multiplizieren im Exponenten ermöglicht.
- Anwendung: Wir zeigen, wie man mit den von multilinearen Abbildungen bereitgestellten Multiplikationen im Exponenten bestimmte in der Kryptographie verwendete Funktionen (*lossy trapdoor functions*) kompakt darstellen kann.

- Effizienz: Wir beschreiben eine Transformation, durch die man bestimmte Kryptosysteme, welche multilineare Abbildungen benötigen, um ein Vielfaches effizienter implementieren kann.

Chapter 1

Introduction

In the 1970s, the invention of the first public-key cryptosystems launched a whole new area of cryptographic research. Public-key cryptosystems allow (two) parties to establish a secret among them only from publicly available information. This removes the burden of first establishing a secret through different means from the already long known secret key cryptosystems. Triggered by the seminal works of Merkle [92] and Diffie and Hellman [39], countless public key constructions for key exchange, encryption, digital signatures and more cryptographic primitives have been invented. The growing list includes famous examples such as the Diffie–Hellman Key Exchange [39] and the RSA cryptosystem [101], both being essential components of widely used protocols such as TLS, S/MIME and SSH. Established notions of security for public key constructions include semantic security for encryption and existential unforgeability for digital signatures. It is well known that proving a construction to achieve one of these notions would imply solving a problem that has been depriving theorists from sleep since the middle of the last century [1]. Meanwhile, it is common practice to prove the security of a cryptographic construction with respect to an assumption. Such an assumption usually states computational hardness of a mathematical problem. For example, security of the Diffie–Hellman Key Exchange relies on the hardness of computing the discrete logarithm in a cyclic group, while the RSA cryptosystem can only be secure when factoring large numbers is hard. Besides these two main directions, lattice-based cryptography relies on the hardness of computational problems in lattices. And of course, one can always introduce more specific assumptions, e.g., from the field of algebraic coding theory [88], or hit the big score by successfully basing security on NP-complete problems (e.g., [91]). This thesis, however, is in the setting where security is based on the hardness of computing discrete logarithms.

1.1 Cryptography Based on the Discrete Logarithm Problem

DISCRETE LOGARITHMS. In cryptography, talking about discrete logarithms requires a cyclic group G of finite order n with generator g. Now the discrete logarithm of an element $h \in G$ with respect to basis g is (the uniquely determined) $a \in \mathbb{Z}_n$ such that $g^a = h$. In other words, computing the discrete logarithm reverses exponentiation. This computation, called DLOG, can be relatively easy. For example, if G is the additive group $\mathbb{Z}_n$ itself, one can use the Extended Euclidean Algorithm to solve DLOG in polynomial time, i.e., $\mathbf{O}(\log_2(n)^2)$. However, no efficient algorithm for

[1] The P versus NP problem, see, e.g., `http://www.claymath.org/millennium-problems`

computing DLOG in any group is known. The best known algorithms in, e.g., multiplicative subgroups of finite fields, are Index Calculus algorithms, which have sub-exponential runtime. Even worse, there are elliptic curves[2] where we cannot apply Index Calculus. Here, the best known algorithms run in $\mathbf{O}(\sqrt{n})$, where n is the size of the group[3]. While the intractability of DLOG regarding these two families of groups might be annoying for algorithmicians, cryptographers quite like it[4].

THE NEED FOR STRONGER ASSUMPTIONS. Unfortunately, groups with a presumably hard DLOG problem alone are not sufficient for cryptographic purposes. DLOG, as it turns out, is often too hard to admit a reduction from the security of a cryptographic construction. Security can thus only be shown under stronger assumptions, leaving hardness of DLOG to be a necessary but not sufficient condition. Some of these stronger (but nowadays standard) assumptions historically arose from stating the security of a cryptographic construction. The probably most famous example is the computational Diffie–Hellman assumption (CDH), which demands that given (only the public keys of two users executing the Diffie–Hellman Key Exchange protocol) g^a, g^b it is hard to compute (the shared key of the Diffie–Hellman Key Exchange) g^{ab}. It is straightforward to see that this assumption can only hold in groups where DLOG is hard, while the converse is not known to be true. The vast majority of cryptographic constructions in the discrete-log type setting rely on assumptions stronger than CDH. All of them exploit the fact that computing or even recognizing products "in the exponent" is hard.

1.2 Advances from Mathematics: Pairings

A pairing is an efficiently computable nontrivial bilinear map e that maps two cyclic groups G_1, G_2 to another cyclic group G_T, where potentially $G_1 = G_2$ (in this case we refer to e as *symmetric*, otherwise as *asymmetric*). G_1 and G_2 are often referred to as *bilinear groups*. Until a few years ago, the only known constructions were the Weil and the Tate pairing and variations thereof. Here, G_1 and G_2 are elliptic curves and G_T is a finite field. What makes these maps interesting for cryptographic purposes is the bilinearity property

$$e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}.$$

A pairing thus gives us something close to multiplication "in the exponent" - but what is the point of having the product of the exponents *in a different group*? Let us have a closer look at this map.

[2] The points of an elliptic curve form a group together with an operation that is probably best explained by looking at a picture, e.g., in [111], on page 56.

[3] We note that there are methods that exhibit specific properties of the group and can run faster in certain scenarios. For example, the Pollard Rho Algorithm runs in $\mathcal{O}(k\sqrt{p})$, where p is the largest prime factor of the group order and k its multiplicity.

[4] At least as long as they do not think about the dark clouds on the horizon, where there are quantum computers that will render all cryptography based on factorization and discrete logarithms insecure [109].

PAIRINGS AS PROBLEM SOLVERS. In this thesis, we assume that G_T is neither G_1 nor G_2[5]. Bilinearity refutes injectivity of e so that we do not have an inverse map. It is not obvious, even not in the symmetric case, how to use a pairing to multiply exponents in the source groups. Nonetheless, we can use it to multiply exponents in another group. This directly implies that certain cryptographic assumptions, such as hardness of recognizing products "in the exponent" (the decisional Diffie–Hellman Assumption, or DDH), become easy in the presence of a symmetric pairing. Moreover, Menezes, Okamoto and Vanstone gave a polynomial reduction (often referred to as the MOV attack, [90] of DLOG on an elliptic curve with a symmetric pairing to DLOG in the finite field G_T. In certain cases[6], i.e., when the order of G_T is comparably small, the sub-exponential Index Calculus methods for solving DLOG in G_T outperform the square-root algorithms for solving DLOG in the elliptic curves. This renders certain classes of elliptic curves useless for cryptographic constructions whose security is based on the hardness of DLOG.

PAIRINGS IN CRYPTOGRAPHIC CONSTRUCTIONS. After being used only to scratch candidates from the pool of groups used in cryptography for nearly a decade, pairings received the attention of cryptographers again. In the year 2000, Joux used them in the first ever known construction of non-interactive key exchange among three users [76]. The construction is a generalization of the Diffie–Hellman Key Exchange. It advises a user to compute a shared key $K \in G_T$ from two public keys $g^a, g^b \in G$ and a secret key $c \in \mathbb{Z}_p$ as $K := e(g^a, g^b)^c = e(g, g)^{abc}$ by using a symmetric pairing $e : G \times G \to G_T$. From that point on, pairings were rehabilitated. Within the next years, they led to a series of breakthrough instantiations of so far only theoretical concepts such as identity-based encryption [12] and key agreement [106], non-interactive zero-knowledge proof systems [63], short signatures [15], and attribute-based encryption [105, 62].

1.3 Generalizing Pairings: Multilinear Maps

MULTILINEAR MAPS. One can generalize the Diffie–Hellman Key Exchange even more to include up to $k+1$ users for any $k \in \mathbb{N}$. To combine k exponents from cyclic groups $G_1, \ldots, G_k$ with a resulting shared key in a target group G_T, we need a mapping between $G_1, \ldots, G_k$ and G_T that is linear in each component:

$$e(g_1^{a_1}, \ldots, g_k^{a_k}) = e(g_1, \ldots, g_k)^{\prod_{i=1}^{k} a_i}.$$

These so-called *multilinear maps* (or k-linear map in the above case) were introduced in [16], together with desirable applications such as the generalized Diffie–Hellman Key Exchange. To be of any interest for cryptographic purposes, it should at least be hard to compute DLOG in the source and target groups of e. Even more, for instantiating the $k+1$-user Diffie–Hellman Key Exchange protocol, we require a map where computing (or, for stronger security, recognizing) products of $k+1$ exponents is hard in either the source group and target groups of e.

[5] We note that a very recent line of research, starting with [83], investigates the existence and cryptographic relevance of *self-bilinear maps*, where source and target groups are identical.

[6] Considering the latest progress in computing DLOG in finite fields, [65] gives an overview of which curves should be avoided due to the MOV attack.

Simple as they are, multilinear maps turned out to be an extremely powerful primitive in cryptography. We refer the reader to the upcoming section for applications. In fact, as it is often the case with powerful tools, researchers even have started doubting their existence at all. Currently, the question is: Can we make them or can we break them?

GRADED ENCODING SCHEMES. Very recently, the notion of *graded encoding systems* (GES) was introduced by [51]. Roughly speaking, a GES is an approximate version of groups equipped with a multilinear map, but instead of groups one has sets of non-unique encodings of exponents. In most cases[7], applications relying on multilinear maps can be easily modified to use a GES instead. Besides, a GES offers the additional functionality of multiplying two exponents (with the result lying in an intermediate set, often referred to as a "level") instead of providing only a way of multiplying k exponents at once, as with a k-linear map. We mention cryptographic applications exploiting this *graded* property of a GES as the chapter unfolds.

RESEARCH ON MULTILINEAR MAPS. From what we have seen so far, the following natural questions arise.

- *Constructions:* Can we construct a GES (or multilinear map)?
- *Applications:* Can we find new ways to use a GES (or multilinear map) for cryptographic purposes?
- *Efficiency:* Can we speed up cryptographic constructions that make use of a GES (or multilinear map)?

1.4 State of the Art

CONSTRUCTIONS. The first candidate construction of a GES was proposed by Garg, Gentry and Halevi [51]. Their scheme (dubbed GGH) works with ideal lattices and uses growing noise in encodings to argue about an upper bound on multiplications in the exponent. Unfortunately, this noise also prevents the operation in each source set from being closed. It also hinders efficient procedures, e.g., for recognizing encodings of zero or extracting unique representations from encodings, both required by many cryptographic constructions. In GGH, eliminating the noise can only be done in the target set by extracting a unique representation, but this comes at a price: one can no longer compute with extracted elements. To enable extraction or zero-testing, GGH proposes to publish additional information about the secret values of the scheme. Unfortunately, this makes GGH prone to attacks (dubbed zeroizing attacks, [51] that turned out to be quite devastating [59]. Ever since, there is an ongoing flurry of fixes and attacks, where [32, 24, 57, 34, 31] is not an exhaustive list.

But what about applications that neither require zero-testing nor extraction, like, e.g., indistinguishability obfuscation ($i\mathcal{O}$) [6]? This cryptographic primitive computes a circuit from one of two input/output-equivalent circuits but does not reveal

[7] See Section 4.1.1 or [119] for applications where it is not known how to implement them with a generic GES.

which circuit was used. It has been proven to be an extremely powerful tool in cryptographic constructions [104]. Recently, a candidate construction was proposed [52] that uses a weaker form of multilinear maps without zero-testing and extraction. One can thus hope to implement it using GGH without the fatal publishing of additional information. [93] shows that there are still attacks, against which [52] was immunized shortly after [55]. As of today, it seems to be possible to securely instantiate $i\mathcal{O}$ with GGH. Obviously, more investigations are required to clarify which security GGH provides in each of the aforementioned cases.

In the meantime, other constructions of GESs have been proposed. [35] uses techniques similar to those of GGH but works over the integers instead of using ideal lattices. Unfortunately, zeroizing attacks seem to be even more devastating for this candidate and its subsequent version [37], resulting in a total break of the scheme [59, 21, 36, 30, 29].

Taking a completely different approach, the work of Albrecht et al. [3] aims at providing multilinear maps in a setting where the underlying groups have randomized encodings. In a nutshell, they build a group with a multilinear map from a cyclic group with a hard Diffie–Hellman-like problem by adding auxiliary information to each element[8]. To show that various cryptographic assumptions hold in their construction, [3] makes generic use of $i\mathcal{O}$. Relying on heavy assumptions such as $i\mathcal{O}$ is the main drawback of their construction, but still their result reduces the task of finding a multilinear map to the task of finding an indistinguishability obfuscator. As argued above, this might be a problem that is easier to solve.

APPLICATIONS. Besides being used for constructing $i\mathcal{O}$ [52, 119], graded encoding schemes have several other important applications in cryptography. Among them are the first ever known constructions of so far only theoretical concepts, like multi-user non-interactive key exchange (as explained above), attribute-based encryption for general circuits [53], constrained pseudo random functions for general circuits [19], full domain hash [75] and multi-input functional encryption [61, 14][9]. Applications that can be implemented with both multilinear maps and a GES are witness encryption [54], low overhead broadcast encryption [20] and programmable hash functions [48].

EFFICIENCY. When it comes to implementing any application relying on multilinear maps, even in the bilinear case the bottleneck in efficiency is the computation of the map. To get an insight into possible measures of tweaking this, we first look at existing efficiency improvements for pairings. We identify the following three kinds of improvements: we can a) find a map that can be computed faster by construction, or b) find better parameters (i.e., elliptic curves in the case of pairings) or, finally, c) optimize the algorithms for evaluating the map. For pairings, the most notable work in a) includes studies of optimality of pairings [113] or optimizing an existing pairing, the Eta Pairing, to be easier to compute [72]. Better parameters can be found by looking into properties of elliptic curves, e.g., by choosing BLS12 curves for computing asymmetric pairings with 192-bit security [4]. Another line of research aims at replacing groups of composite order with (several copies of) groups of prime order [45, 87]. This enhances performance since groups of prime order can be

[8] See Section 3.3 for a summary of their construction.

[9] In a recent result, multi-input functional encryption is achieved from pairings [1].

chosen significantly smaller (e.g., 256 bit size) than groups of composite order (e.g., 2048 bit size) to resist factorization attacks. Of course, one then has to argue how a pairing on composite-order groups can be emulated using a pairing on prime-order groups. And finally, one can speed up Miller's Algorithm [5], which has been the best known way to evaluate pairings since its invention in 1986.

For multilinear maps, the focus currently lies on finding new constructions and dealing with attacks on existing ones. There is thus much room for improvements in all aforementioned cases. Regarding a), GGHLite [82] was proposed, followed by further optimizations in [2][10]. For b), one strategy is to apply the aforementioned transformations from composite order to prime order. Unfortunately, none of the existing approaches for pairings scales to the multilinear case (see Section 5.4 for the actual numbers).

1.5 Contributions of this Thesis

In this thesis, we shed more light on all of the research questions posed above. Our contributions are threefold. First, we give a construction of a GES that is, as of today and to our knowledge, the only construction in which certain multilinear assumptions can hold. Second, we use multilinear maps to make a cryptographic primitive called *lossy trapdoor function* more efficient. And finally, we give a new composite-to-prime-order transformation. Besides being more efficient than existing transformations already in the case of pairings, our construction is the only one that can be applied to composite-order cryptosystems using multilinear maps (or a GES, respectively). Let us discuss these contributions in more detail below.

Constructions

In our first contribution, we propose a new candidate graded encoding scheme in [43]. Technically, we adopt the approach of [3] and equip a cyclic group $G = \langle g \rangle$ with auxiliary information. More detailed, an encoding of a is of the form

$$(g^a, \texttt{auxiliary information}).$$

In [3], the auxiliary information stashes away a linear polynomial that evaluates to a at a secret fixed point. To achieve a graded property, we view exponents not only as linear polynomials but polynomials of higher degrees. Tying the maximum degree to the level gives us different sets of encodings in each level. Polynomial multiplication in the exponent is then a bilinear map in each two of these sets. Of course, we have to find means to enable multiplication in the exponent without giving away too much, since we still want to restrict the number of possible multiplications to a specific value. For this, like [3], we use $i\mathcal{O}$.

We want to highlight several drawbacks, benefits and implications of our construction. Obviously, the main drawback is the dependence on $i\mathcal{O}$. Additionally, we were not able to give an efficient procedure for testing validity of encodings (in the sense that the auxiliary information is honestly generated). Turning to benefits, our

[10] Implementations can be found in, e.g., 5Gen [84], a software stack that aims at providing libraries for applications using multilinear maps.

construction admits efficient algorithms for several tasks. E.g., testing equality of encodings (simply compare the first entry) or extracting a canonical form (simply cut the auxiliary information) both come essentially for free. Our GES is the first to enable efficient computation with unique representations of encodings in the target set, which in our construction is simply the cyclic group G. Regarding security, we are able to give a clean reduction from a multilinear variant of DDH (called MDDH) to a standard cryptographic assumption called SDDH in G (using $i\mathcal{O}$ and a bilinear group)[11]. Regarding the attacks from the literature on existing GES candidates, we are optimistic that none of these techniques apply to our candidate, since we do not use any noise. Furthermore, like [3] for the case of multilinear maps, our construction reduces the task of finding a GES to the task of finding $i\mathcal{O}$.

Applications

A family of lossy trapdoor functions (LTDF) is a collection of functions parametrized by two fixed values for input size and lossiness. A function from this collection is either injective or not injective. The latter functions, also called the *lossy* ones, loose as many bits of information about the input as the lossiness parameter dictates. The key property now is that injective and lossy functions are indistinguishable.

LTDFs are a powerful primitive with many applications in cryptography. To mention just one example, LTDFs can be used to build IND-CCA secure encryption [99, 94]. We refer the reader to Chapter 4 for a more elaborated list of applications.

LTDFs themselves are known to exist under various cryptographic assumptions. In the DLOG-based setting, the first constructions had a public description of $\mathbf{O}(n^2)$ group elements [99, 46, 115] where n is the (bit)size of the function's input. Later, this was improved to $\mathbf{O}(n)$ group elements using a pairing [23]. Unfortunately, the techniques of [23] do not seem to generalize to the multilinear case to achieve further improvements. This is where our work begins. In [73], we give a technique to further compress the description of DLOG-based LTDFs to $\mathbf{O}(n^{2/k})$ using a k-linear map. A nice side effect is that, already for the case of pairings ($k = 2$), our LTDF is shorter than the one from [23].

Unfortunately, it seems to be hard to implement our LTDF, as well as LTDFs in general, with GGH. The reason is that, roughly speaking, noise might carry information about the input, making it harder to argue for lossiness of a function. To implement our compact LTDF, we can choose the noise-free multilinear map and GES constructions from [3] and [43], respectively[12].

Efficiency

In [71], we propose a new transformation that renders cryptographic constructions in composite-order groups using a k-linear map more efficient. Looking at these

[11] Using the same set of assumptions, namely SDDH, $i\mathcal{O}$ and a bilinear group, we can also show hardness of an assumption about deciding the rank of an encoded matrix (called RANK).

[12] We mention here that the assumptions required by our constructions, namely either DDH or the k-Linear Assumption, are not known to hold in [3], [43]. Considered more optimistically, they are also not known to be false and both constructions already provide security regarding RANK, which is only slightly weaker than the k-Linear Assumption.

constructions (e.g., [60, 13, 18, 79, 114, 89], we see that security often relies on the fact that membership in certain subgroups is hard to recognize. Often, applications using multilinear maps in composite-order groups (e.g., [114, 89] exploit properties such as orthogonality of subgroups (in the sense that pairing two subgroups of different order yields the neutral element). The technical difficulty is to make sure that our emulated composite-order groups and, especially, the emulated map on these groups provide all of these features.

All existing approaches from the literature [45, 89, 108, 107, 85, 86] are designated for the case of pairings. To emulate a composite-order group, they take several (say, n) copies of a group of prime order p. Elements are now vectors of prime-order group elements. Regarding security, recognizing subgroup elements now corresponds to recognizing linear subspaces of $\mathbb{Z}_p^n$. A pairing can be obtained by multiplying vectors "in the exponent" using the underlying prime-order pairing. All existing transformations multiply vectors using a tensor product, resulting in a large number of prime-order pairing evaluations. Again, we find ourselves in a situation where existing techniques for pairings do not work for the multilinear case. Namely, generalizing any of the existing transformations to the k-linear case (where we have to multiply k vectors instead of 2) results in an exponential number of prime-order pairing evaluations.

In our transformation, we follow the same approach but switch to a more efficient vector multiplication. In a nutshell, we interpret vectors as coefficient vectors of a polynomial and use polynomial multiplication as a k-linear map. We can then apply existing techniques to speed up polynomial multiplication in the exponent and thus reduce the number of required prime-order map evaluations to k^2. Our transformation is already the most efficient one in the bilinear case and results in, e.g., implementations of Groth-Sahai proof systems [63] that are roughly two times faster as before.

1.6 Other Results

I also investigated cryptographic topics that are not closely related to multilinear maps and thus do not appear in this thesis. All of them are in the area of public key cryptography. The first result considers a new security notion for non-interactive key exchange (e.g., the aforementioned Diffie–Hellman Key Exchange). Here, non-interactivity is in general achieved by assuming the existence of a public key infrastructure (PKI). The second result, roughly speaking, is about avoiding the tedious update process of a PKI that, in practice, can be huge. Is a PKI lost if there is an attack on the cryptosystem? We identify properties of cryptosystems that help us in rescuing the system without having to update the whole PKI. Short abstracts of both the aforementioned results follow.

Universally Composable Non-Interactive Key Exchange

In [47], we give a new security notion for non-interactive key exchange (NIKE) in the Universal Composability framework (UC). This framework supports a modular analysis of protocols consisting of several phases. Regarding NIKE, e.g., there are two phases: the establishment of a public key infrastructure (PKI) and the actual

computation of the shared key between two parties. But what security does the shared key provide if a malicious PKI is used? Existing game-based security notions for NIKE take this into account by admitting adversaries that pins "bad" public keys on the PKI. Unfortunately, this results in a tedious game and complicates the security analysis of NIKE protocols. Our new notion in the UC framework improves on this by splitting up the protocol according to its phases and analyzing the security of each phase by itself. This simplifies the analysis of NIKE protocols and gives us modularity: *any* UC-secure protocol realizing phase one (distribution of the PKI) combined with *any* UC-secure protocol realizing phase two (computation of the shared key) results in a UC-secure NIKE protocol. Moreover, the UC framework guarantees composability with arbitrary other protocols, including parallel executions of the protocol itself.

Besides defining the new security notion by giving ideal functionalities for each phase of a NIKE, we investigate relations to existing game-based notions. We find that game-based security of NIKE is equivalent to static security in the UC framework (where static means that the adversary may not corrupt any parties after the protocol was started). We also give evidence that a non-static (i.e., adaptive) notion is hard to achieve without using hash functions modeled ideally as random oracles.

Reconfigurable cryptography: A flexible approach to long-term security

In [74], we define the concept of *reconfigurable cryptography*. Reconfigurable here means that there is a procedure that increases the security of a cryptosystem on-the-fly. For practical purposes, we are only interested in fast reconfiguration procedures. Faster than, e.g., the process of updating a whole PKI used by the cryptosystem. For this, a reconfigurable cryptosystem uses a common reference string (CRS) between all participants that can be updated (and downloaded) in a few seconds. A subtlety lies in the level of security that we want to improve: since computing the secret keys from the PKI is an upper bound on the overall security of the system (we call this the "long-term" security), we need to introduce an additional parameter for the "short-term" security that we want to increase on-the-fly. We are only interested in reconfigurable schemes where reconfiguration actually increases the short-term security and where we do not have an a priori bound on the number of possible reconfigurations.

Besides defining reconfigurable cryptosystems and their security, we give three constructions: 1) A reconfigurable encryption scheme that admits reconfiguration in the sense that the cryptosystem is transferred to another (possibly larger) group. This is a nice form of reconfiguration, but also hard to achieve: we did not manage to get it from standard cryptographic assumptions and have to rely on $i\mathcal{O}$ instead. 2) We give a reconfigurable encryption scheme where, upon reconfiguration, short-term security relies on a weaker assumption. More detailed, short-term security relies on the k-SCasc assumption (see 2.4 for a definition) that is known to imply $k+1$-SCasc for any $k \in \mathbb{N}$. And 3), we propose a reconfigurable signature scheme with similar reconfiguration properties from non-interactive zero-knowledge proof systems.

1.7 Organization

Chapter 2 lists the basic definitions we work with. Afterwards, we elaborate on our construction of a GES in Chapter 3. Our construction of a LTDF can be found in Chapter 4. Chapter 5 then explains our composite-to-prime-order transformation. We finish with some concluding remarks in Chapter 6.

Own Publications

Conference papers

- Julia Hesse, Dennis Hofheinz, and Daniel Kraschewski. Lossy trapdoor functions with compact keys from multilinear maps. Unpublished manuscript.
- Pooya Farshim, Julia Hesse, Dennis Hofheinz, and Enrique Larraia. Graded encoding schemes from obfuscation. Unpublished manuscript.
- Julia Hesse, Dennis Hofheinz, and Andy Rupp. Reconfigurable cryptography: A flexible approach to long-term security. In *Eyal Kushilevitz and Tal Malkin, editors, TCC 2016-A, Part I, volume 9562 of LNCS*, pages 416-445. Springer, Heidelberg, January 2016.
- Eduarda S. V. Freire, Julia Hesse, and Dennis Hofheinz. Universally composable non-interactive key exchange. In *Michel Abdalla and Roberto De Prisco, editors, SCN 2014, volume 8642 of LNCS*, pages 1-20. Springer, Heidelberg, September 2014.
- Gottfried Herold, Julia Hesse, Dennis Hofheinz, Carla Ràfols, and Andy Rupp. Polynomial spaces: A new framework for composite-to-prime-order transformations. In *Juan A. Garay and Rosario Gennaro, editors, Crypto 2014, Part I, volume 8616 of LNCS*, pages 261-279. Springer, Heidelberg, August 2014.

Others

- Julia Rohlfing. Paarungen auf elliptischen Kurven und ihre Anwendung in der Kryptografie. Diplomarbeit, Universität Karlsruhe (TH), Germany, April 2009.

Chapter 2

Preliminaries

2.1 Notation

The security parameter will be denoted by $\lambda \in \mathbb{N}$ and we assume that, if not stated explicitly, it is implicitly given to all algorithms in the unary representation 1^λ. By an algorithm we mean a stateless Turing machine. For a probabilistic algorithm $\mathcal{A}$, $y \leftarrow \mathcal{A}(x;r)$ denotes the process of running $\mathcal{A}$ on input x and with randomness $r \in \mathcal{R}_\mathcal{A}$, and assigning y the result. We write $y \leftarrow \mathcal{A}(x)$ for $y \leftarrow \mathcal{A}(x;r)$ with uniform r. If $\mathcal{A}$'s running time is polynomial in λ, then $\mathcal{A}$ is called probabilistic polynomial-time (PPT). For a finite set $\mathcal{S}$, we denote its cardinality by $|\mathcal{S}|$ and the process of sampling s uniformly from $\mathcal{S}$ by $s \xleftarrow{R} \mathcal{S}$. Throughout this thesis $\perp$ denotes a special error symbol, and $\mathrm{poly}(\cdot)$ stands for a fixed (but unspecified) polynomial.

We call a function $\eta : \mathbb{N} \to \mathbb{R}$ *negligible* if for all $c \in \mathbb{N}$ there is a $\lambda_0 \in \mathbb{N}$ such that for all $\lambda > \lambda_0$ it holds that $|\eta(\lambda)| < \frac{1}{\lambda^c}$. We denote the set of all negligible functions by NEGL. We write $D_0 \overset{c}{\approx} D_1$ to denote computational indistinguishability of two distributions D_0, D_1, meaning that no efficient PPT algorithm can decide with non-negligible success if an element x is drawn from D_0 or D_1. Statistical indistinguishability is denoted by $\overset{s}{\approx}$.

We use multiplicative notation for groups, except for Chapter 5, where we use additive notation to visually distinguish between the group operation and polynomial multiplication. Let G be a cyclic group of order ord generated by g. If the generator g is clear from the context, then by $[a] := g^a$ we denote the *implicit representation* of $a \in \mathbb{Z}_{\mathrm{ord}}$ in G. To distinguish between implicit representations in the domains G, G_i and the target group G_T of a multilinear map we use $[\cdot]$, $[\cdot]_i$ and $[\cdot]_T$, respectively. More generally, we also define such representations for sets $H \subset \mathbb{Z}_{\mathrm{ord}}^n$ by $[H] := \{[a] \mid a \in H\} \subset G^n$.

2.2 Vectors and Matrices

We will write matrices in bold capital letters, e.g., $\mathbf{M}$, and denote vectors with an overhead arrow, e.g., $\vec{x}$. Given any matrix $\mathbf{M}$, we denote its determinant by $\det(\mathbf{M})$, its rank by $\mathrm{rank}(\mathbf{M})$ and its trace by $\mathrm{tr}(\mathbf{M})$. We denote the identity matrix by I, or by I_n respectively if we want to highlight that it is of size $n \times n$. We use 0 and 0_n for the all-zero matrix of size $n \times n$ accordingly.

We generalize the implicit notation in groups to vectors and matrices. Namely, if G is a group of order ord and $\vec{x} \in \mathbb{Z}_{\mathrm{ord}}^n$ and $\mathbf{M} \in \mathbb{Z}_{\mathrm{ord}}^{m \times n}$ we write $[\vec{x}] := ([x_i])_i \in G^n$ and $[\mathbf{M}] := ([m_{i,j}])_{i,j} \in G^{m \times n}$. We define a special matrix-vector product $*$ by $[\mathbf{M}] * \vec{x} = \mathbf{M} * [\vec{x}] := [\mathbf{M}\vec{x}]$ and $[\vec{x}] * \mathbf{M} = \vec{x} * [\mathbf{M}] = [\vec{x}\mathbf{M}]$, where $\mathbf{M}$ and $\vec{x}$ are both

defined over $\mathbb{Z}_p$ with suitable dimensions. (Note that $*$ is efficiently computable.)

Furthermore, we will sometimes identify $\vec{f} \in \mathbb{Z}_{\text{ord}}^n$ with the coefficients of a polynomial f in some space V with respect to a (fixed) basis $\mathfrak{q}_0, \ldots, \mathfrak{q}_{n-1}$ of V, i.e., $f = \sum_{i=0}^{n-1} f_i \mathfrak{q}_i$ (e.g., $V = \{f \mid f \in \mathbb{Z}_{\text{ord}}[X], \deg(f) < n\}$ and $\mathfrak{q}_i = X^i$). In this case we may also write $[f] := [\vec{f}]$.

We will use the mathematical concept of the *tensor product* of two matrices (also called the *Kronecker product*). One important property of the Kronecker product is that the size of the result is the product of the sizes of the two inputs. This will be helpful for the compression of public keys containing matrices. While the classical Kronecker product combines elements using the group operation, we will give a generalized definition which implies the classical definition.

Definition 2.2.1 (Generalized Kronecker product)**.** *For $m, n, q, r \in \mathbb{N}$ let $\mathbf{M} \in G_1^{m \times n}, \mathbf{M}' \in G_2^{q \times r}$ be defined over groups G_1, G_2 and $\phi : G_1 \times G_2 \to G_T$ a function, where G_T is also a group. We extend ϕ to matrices of group elements in a componentwise fashion (i.e., $\phi(m, \mathbf{M}')$ for $\mathbf{M}' = (m'_{i,j})$ is the matrix $(\phi(m, m'_{i,j}))$). Then the* generalized Kronecker product $\mathbf{M} \otimes_\phi \mathbf{M}'$ *is defined as*

$$\mathbf{M} \otimes_\phi \mathbf{M}' := \begin{pmatrix} \phi(m_{1,1}, \mathbf{M}') & \phi(m_{1,2}, \mathbf{M}') & \ldots \\ \phi(m_{2,1}, \mathbf{M}') & \phi(m_{2,2}, \mathbf{M}') & \\ \vdots & & \ddots \end{pmatrix} \in G_T^{mq \times nr}.$$

When $G_1 = G_2 = G_T$ with $\phi(g, h) := gh$ we obtain the classical Kronecker product, which will be denoted by $\otimes$. For this case we recall some basic calculation rules which we will mainly use in Section 4.4. For matrices $\mathbf{M}, \mathbf{M}', \mathbf{Q}, \mathbf{Q}'$ over the same field K we have

$\mathbf{M} \otimes (\mathbf{Q} + \mathbf{Q}') = \mathbf{M} \otimes \mathbf{Q} + \mathbf{M} \otimes \mathbf{Q}'$	if $\mathbf{Q}$ and $\mathbf{Q}'$ are of the same size
$\mathbf{M}\mathbf{M}' \otimes \mathbf{Q}\mathbf{Q}' = (\mathbf{M} \otimes \mathbf{Q}) \cdot (\mathbf{M}' \otimes \mathbf{Q}')$	if the matrix products $\mathbf{M}\mathbf{M}'$ and $\mathbf{Q}\mathbf{Q}'$ are defined
$\det(\mathbf{M} \otimes \mathbf{M}') = \det(\mathbf{M})^{n'} \cdot \det(\mathbf{M}')^n$	if $\mathbf{M}$ is of size $n \times n$ and $\mathbf{M}'$ is of size $n' \times n'$
$\text{rank}(\mathbf{M} \otimes \mathbf{M}') = \text{rank}(\mathbf{M}) \cdot \text{rank}(\mathbf{M}')$	

2.3 Multilinear Maps and Graded Encoding Schemes

Definition 2.3.1 (Multilinear map)**.** *Let $k, \text{ord} \in \mathbb{N}$ and $G_1, \ldots, G_k, G_T$ be cyclic groups of order* ord*. A* k-linear map $e : G_1 \times \cdots \times G_k \longrightarrow G_T$ *is a map with the following properties:*

- ***Multilinearity:*** *e is k-linear in the sense that for all $i \in [k]$ and $\alpha \in \mathbb{Z}_{\text{ord}}$ we have*
$$e(g_1, \ldots, g_i^\alpha, \ldots, g_k) = e(g_1, \ldots, g_k)^\alpha.$$
- ***Non-Degeneracy:*** *for each set of generators $g_1, \ldots, g_k$ of the groups $G_1, \ldots, G_k$, the element $e(g_1, \ldots, g_k)$ generates G_T.*

If $G_1 = \cdots = G_k$ we say e is a symmetric *map, otherwise e is called* asymmetric*. We call $(G_1, \ldots, G_k, G_T, e, \mathrm{ord})$ a k-linear group.*

Definition 2.3.2 (k-linear group generator)**.** *We define a k-linear group generator to be an algorithm $\mathcal{G}_k$ that gets as input $\lambda \in \mathbb{N}$ and outputs tuples of the form*

$$(k, G_1, \ldots, G_k, G_T, e, \mathrm{ord}, g_1, \ldots, g_k, g_T) \leftarrow \mathcal{G}_k(\lambda),$$

where $G_1, \ldots, G_k, G_T$ are descriptions of cyclic groups with $\mathrm{ord} \in \mathbb{N}$, $\lceil \log_2(\mathrm{ord}) \rceil = \lambda$, g_i generates G_i ($i \in [k]$), g_T generates G_T and $e : G_1 \times \cdots \times G_k \to G_T$ is a k-linear map. We call $\mathcal{G}_k(\lambda)$ symmetric *if $G_1 = \ldots = G_k$, and we refer to it as a* prime-order *group generator it only outputs groups of prime order.*

In the case $k = 1$ we will not need the map and define the output of the group generator $\mathcal{G}_1(\lambda)$ to be the tuple (G, ord, g).

GRADED ENCODING SCHEMES. We recall (a slight variant of) the definition of graded encoding schemes from Garg, Gentry and Halevi (GGH) [51]. To this end, we start by defining graded encoding systems. Notice the small but fine difference in the name: a graded encoding *system* is a mathematical structure, while a graded encoding *scheme* is this very structure together with a set of efficient algorithms for various tasks.

Definition 2.3.3 (Graded Encoding System)**.** *Let R be a (non-trivial) commutative ring and $S := \{S_i^{(a)} \subset \{0,1\}^* \ : \ a \in R, 0 \leq i \leq \kappa\}$ a system of sets. Then (R, S) is called a κ-*graded encoding system *if the following conditions are met.*

1. *For each* level *$i \in \{0, \ldots, \kappa\}$ and for any $a_1, a_2 \in R$ with $a_1 \neq a_2$ we have that $S_i^{(a_1)} \cap S_i^{(a_2)} = \emptyset$.*

2. *For each level $i \in \{0, \ldots, \kappa\}$, the set $\{S_i^{(a)} \ : \ a \in R\}$ is equipped with a binary operation "$+$" and a unary operation "$-$" such that for all $a_1, a_2 \in R$ and every $u_1 \in S_i^{(a_1)}, u_2 \in S_i^{(a_2)}$ it holds that*

$$u_1 + u_2 \in S_i^{(a_1+a_2)} \quad \text{and} \quad -u_1 \in S_i^{(-a_1)}.$$

   Here, $a_1 + a_2$ and $-a_1$ denote addition and negation is R.

3. *For each two levels $i, j \in \{0, \ldots, \kappa\}$ with $i + j \leq \kappa$, there is a binary operation "$\times$" such that for all $a_1, a_2 \in R$ and every $u_1 \in S_i^{(a_1)}, u_2 \in S_j^{(a_2)}$ it holds that*

$$u_1 \times u_2 \in S_{i+j}^{(a_1 \cdot a_2)}.$$

   Here, $a_1 \cdot a_2$ denotes multiplication in R.

The difference to the GGH definition is that we do not require the operations "$+$" and "$\times$" to be associative or commutative. (Indeed, our upcoming construction does not satisfy these properties.) We are not aware of any applications that require the associativity or commutativity of *encodings*. However, we stress that the operations "$+$" and "$\times$" must respect the ring operations from R. For instance, while we may

have $(u_1 + u_2) + u_3 \neq u_1 + (u_2 + u_3)$ for some $u_i \in S_j^{(a_i)}$, both the left-hand and the right-hand sides lie in $S_j^{(a_1+a_2+a_3)}$.

In the setting of graded encoding systems, we refer to an element $a \in R$ as an *exponent* and a bit string $u \in S_i^{(a)}$ as an *encoding* of a. Further, we write $S_i := \bigcup_{a \in R} S_i^{(a)}$ for the set of all level-i encodings.

We now define graded encoding *schemes* by introducing explicit algorithms for manipulating encodings of a graded encoding system.

Definition 2.3.4 (Graded Encoding Scheme (GES))**.** *Let (R, S) be a κ-graded encoding system. A* graded encoding scheme (GES)

$$\Gamma = (\mathsf{Setup}, \mathsf{Eq}, \mathsf{Add}, \mathsf{Mult}, \mathsf{Sam}, \mathsf{Ext})$$

associated to (R, S) consists of the following PPT *algorithms.*

$\mathsf{Setup}(1^\lambda, 1^\kappa)$*: On input the security parameter 1^λ and the (multi)linearity 1^κ, it outputs parameters of Γ (which are assumed to be provided to all other algorithms). We note that this algorithm runs in time $\mathrm{poly}(\lambda)$ as long as κ is polynomial in λ.*

$\mathsf{Eq}_i(h_1, h_2)$*: For $i \in \{0, \ldots, \kappa\}$ and two encodings $h_1 \in S_i^{(a)}$ and $h_2 \in S_i^{(b)}$, this deterministic algorithm outputs 1 if and only if $a = b$ in R.*

$\mathsf{Add}_i(h_1, h_2)$*: This deterministic algorithm performs the "$+$" operation of (R, S) in level i. For $i \in \{0, \ldots, \kappa\}$ and encodings $h_1 \in S_i^{(a_1)}$ and $h_2 \in S_i^{(a_2)}$ this algorithm outputs an encoding in $h \in S_i^{(a_1+a_2)}$.*

$\mathsf{Mult}_{i,j}(h_1, h_2)$*: This deterministic algorithm performs the "$\times$" operation of (R, S). For $i, j \in \{0, \ldots, \kappa\}$ with $i + j \leq \kappa$ and encodings $h_1 \in S_i^{(a_1)}$ and $h_2 \in S_j^{(a_2)}$ this algorithm outputs an encoding in $S_{i+j}^{(a_1 \cdot a_2)}$.*

$\mathsf{Sam}_i(a)$*: For $i \in \{0, \ldots, \kappa\}$ and $a \in R$, this probabilistic algorithm samples an encoding from $S_i^{(a)}$.*

$\mathsf{Ext}_i(h)$*: For $i \in \{0, \ldots, \kappa\}$ and input $h \in S_i$, this deterministic algorithm outputs a bit string. Algorithm Ext_i is required to respect membership in $S_i^{(a)}$ in the sense that it outputs identical strings for any two encodings $h_1, h_2 \in S_i^{(a)}$*

We now define a κ-MLG scheme, a restricted version of a graded encoding scheme where encodings belong to levels 0, 1 and κ only and the Mult algorithm takes κ encodings at level 1 and outputs an encoding at level κ.

SYMMETRIC MLG SCHEMES. A symmetric κ-linear group scheme is a κ-graded encoding scheme associated to (R, S), where (R, S) is defined similarly to a κ-graded encoding system except that $S := \{S_i^{(a)} \subset \{0,1\}^* \ : \ a \in R, i \in \{0, 1, \kappa\}\}$ and the "$\times$" operation is redefined as a κ-ary map that for any $a_1, \ldots, a_\kappa \in R$ and any $u_1 \in S_1^{(a_1)}, \ldots, u_\kappa \in S_1^{(a_\kappa)}$ satisfies

$$u_1 \times \cdots \times u_\kappa \in S_\kappa^{(a_1 \cdots a_\kappa)}.$$

The associated Mult algorithm on inputs $h_i \in S_1^{(a_i)}$ for $i \in \{1, \ldots, \kappa\}$ outputs an encoding in $S_\kappa^{(a_1 \cdots a_\kappa)}$. Algorithms Eq, Add, Sam and Ext are defined analogously and restricted to $i \in \{0, 1, \kappa\}$ only.

2.4 Cryptographic Assumptions in the Discrete-Log based Setting

We list the computational problems whose presumed hardness[1] is the basis for the constructions in this thesis. We start with classical DDH and (stronger) variations thereof, followed by a whole class of Diffie–Hellman-like assumptions called *matrix assumptions*. Finally, we formulate two assumptions in groups equipped with a multilinear map.

DIFFIE–HELLMAN-LIKE ASSUMPTIONS. For all definitions in this paragraph, $\mathcal{G}_1$ denotes the group generator from Definition 2.3.2 that outputs tuples of the form (G, ord, g).

Definition 2.4.1 (Decisional Diffie–Hellman Assumption (DDH))**.** *We say that the* DDH *assumption holds with respect to* $\mathcal{G}_1$ *if*

$$\mathbf{Adv}^{\mathrm{ddh}}_{\mathcal{G}_1,\mathcal{A}}(\lambda) := 2 \cdot \Pr\left[\mathrm{DDH}^{\mathcal{A}}_{\mathcal{G}_1}(\lambda)\right] - 1 \in \mathrm{NEGL},$$

where game $\mathrm{DDH}^{\mathcal{A}}_{\mathcal{G}_1}(\lambda)$ *is shown in Figure 2.1 (top left).*

Definition 2.4.2 (q-SDDH Assumption (q-SDDH) [10, 118])**.** *For* $q \in \mathbb{N}$ *we say that the* q-SDDH *assumption with respect to* $\mathcal{G}_1$ *if*

$$\mathbf{Adv}^{q\text{-sddh}}_{\mathcal{G}_1,\mathcal{A}}(\lambda) := 2 \cdot \Pr\left[q\text{-}\mathrm{SDDH}^{\mathcal{A}}_{\mathcal{G}_1}(\lambda)\right] - 1 \in \mathrm{NEGL},$$

where game q-$\mathrm{SDDH}^{\mathcal{A}}_{\mathcal{G}_1}(\lambda)$ *is shown in Figure 2.1 (top right).*

Definition 2.4.3 (k-Linear Assumption (k-LIN))**.** *We say that the* k-LIN *assumption with respect to* $\mathcal{G}_1$ *if*

$$\mathbf{Adv}^{k\text{-lin}}_{\mathcal{G}_1,\mathcal{A}}(\lambda) := 2 \cdot \Pr\left[k\text{-}\mathrm{LIN}^{\mathcal{A}}_{\mathcal{G}_1}(\lambda)\right] - 1 \in \mathrm{NEGL},$$

where game k-$\mathrm{LIN}^{\mathcal{A}}_{\mathcal{G}_1}(\lambda)$ *is shown in Figure 2.1 (bottom).*

Notice that DDH is equal to 1-LIN. Via re-randomization of the generator of the group, one can show that hardness of q-SDDH implies that of $(q-1)$-SDDH. Similarly, k-LIN implies $(k+1)$-LIN.

Instead of using $\mathcal{G}_1$ as a generator, in all above assumptions we can switch to the multilinear setting by using a symmetric multilinear group generator $\mathcal{G}_k$ instead. Note that the existence of a map and a target group plays no role in stating the assumptions, since all generated elements lie in the source group. Then, DDH and q-SDDH cannot hold if we use a symmetric group generator $\mathcal{G}_n$ with $n \geq 2$. The same holds for k-LIN using a symmetric group generator $\mathcal{G}_n$ where $k < n$.

MATRIX ASSUMPTIONS. For our composite-to-prime-order transformation in Chapter 5 it will be convenient to use a framework for Diffie–Hellman-like assumptions introduced in [42]. In a nutshell, the so-called *matrix assumptions* state that it is hard to recognize whether an (encoded) vector lies in the image of an (encoded) matrix or not. Different distributions of matrices then yield different assumptions. The framework captures many assumptions that are already known in the literature.

[1] As common in cryptography, we use the terms "problem $\mathcal{X}$ is hard" and "the $\mathcal{X}$ assumption" interchangeably.

$\underline{\mathrm{DDH}^{\mathcal{A}}_{\mathcal{G}_1}(\lambda)}$
$\mathcal{PP} \leftarrow\!\!\$\ \mathcal{G}_1(\lambda)$
$b \leftarrow\!\!\$\ \{0,1\}$
$\alpha, \beta, \tau_0 \leftarrow\!\!\$\ \mathbb{Z}_{\mathrm{ord}}$
$\tau_1 \leftarrow \alpha\beta$
$b' \leftarrow\!\!\$\ \mathcal{A}(\mathcal{PP}, [\alpha], [\beta], [\tau_b])$
Return $(b = b')$

$\underline{q\text{-}\mathrm{SDDH}^{\mathcal{A}}_{\mathcal{G}_1}(\lambda)}$
$\mathcal{PP} \leftarrow\!\!\$\ \mathcal{G}_1(\lambda)$
$b \leftarrow\!\!\$\ \{0,1\}$
$\omega, \tau_0 \leftarrow\!\!\$\ \mathbb{Z}_{\mathrm{ord}}$
$\tau_1 \leftarrow \omega^{q+1} \pmod{\mathrm{ord}}$
$b' \leftarrow\!\!\$\ \mathcal{A}(\mathcal{PP}, \{[\omega^i]\}_{i=1}^{q}, [\tau_b])$
Return $(b = b')$

$\underline{k\text{-}\mathrm{LIN}^{\mathcal{A}}_{\mathcal{G}_1}(\lambda)}$
$\mathcal{PP} \leftarrow\!\!\$\ \mathcal{G}_1(\lambda)$
$b \leftarrow\!\!\$\ \{0,1\}$
$\alpha_1, \ldots, \alpha_{k+1}, r_1, \ldots, r_k, \tau_0 \leftarrow\!\!\$\ \mathbb{Z}_{\mathrm{ord}}$
$\tau_1 \leftarrow \alpha_{k+1}(r_1 + \cdots + r_k)$
$b' \leftarrow\!\!\$\ \mathcal{A}(\mathcal{PP}, [\alpha_1], \ldots, [\alpha_{k+1}], [\alpha_1 r_1], \ldots, [\alpha_k r_k], [\tau_b])$
Return $(b = b')$

Figure 2.1: The DDH, q-SDDH and k-LIN security games.

Definition 2.4.4 (Matrix Distributions and Assumptions [42])**.** *Let $n, \ell \in \mathbb{N}$, $n > \ell$. We call $\mathcal{D}_{n,\ell}$ a* matrix distribution *if it outputs (in probabilistic polynomial time, with overwhelming probability) matrices $\mathbf{A} \in \mathbb{Z}_{\mathrm{ord}}^{n\times\ell}$ of full rank ℓ. $\mathcal{D}_{n,\ell}$ is called* polynomially induced *if it is defined by picking $\vec{s} \in \mathbb{Z}_p^d$ uniformly at random and setting $a_{i,j} := \mathfrak{p}_{i,j}(\vec{s})$ for some polynomials $\mathfrak{p}_{i,j} \in \mathbb{Z}_p[\vec{X}]$ whose degrees do not depend on the security parameter. We define $\mathcal{D}_\ell := \mathcal{D}_{\ell+1,\ell}$. Furthermore, we say that the $\mathcal{D}_{n,\ell}$-*Matrix Diffie-Hellman assumption or just $\mathcal{D}_{n,\ell}$ assumption for short *holds relative to the group generator $\mathcal{G}_1$ if for all PPT adversaries $\mathcal{D}$ we have*

$$\mathbf{Adv}_{\mathcal{D}_{n,\ell},\mathcal{G}_1}(\mathcal{D}) = \mathbf{Pr}[\mathcal{D}(\mathcal{PP}, [\mathbf{A}], [\mathbf{A}\vec{w}]) = 1] - \mathbf{Pr}[\mathcal{D}(\mathcal{PP}, [\mathbf{A}], [\vec{u}]) = 1] = \mathrm{NEGL}(\lambda) \ ,$$

where the probability is taken over the output $\mathcal{PP} = (G, \mathrm{ord}, g) \leftarrow \mathcal{G}_1(\lambda)$, $\mathbf{A} \leftarrow \mathcal{D}_{n,\ell}$, $\vec{w} \leftarrow \mathbb{Z}_{\mathrm{ord}}^{\ell}$, $\vec{u} \leftarrow \mathbb{Z}_{\mathrm{ord}}^{n}$ and the coin tosses of the adversary $\mathcal{D}$.

We note that all of the standard examples of matrix assumptions are polynomially induced and further, in all examples we consider in this thesis, the degree of $\mathfrak{p}_{i,j}$ is 1. In particular, we will refer to the following examples of matrix distributions, all for $n = \ell + 1$:

$$\mathcal{SC}_\ell : \mathbf{A} = \begin{pmatrix} -s & 0 & \ldots & 0 & 0 \\ 1 & -s & \ldots & 0 & 0 \\ 0 & 1 & & 0 & 0 \\ \vdots & & \ddots & & \\ 0 & 0 & \ldots & 1 & -s \\ 0 & 0 & \ldots & 0 & 1 \end{pmatrix} \quad \mathcal{L}_\ell : \mathbf{A} = \begin{pmatrix} s_1 & 0 & 0 & \ldots & 0 \\ 0 & s_2 & 0 & \ldots & 0 \\ \vdots & \vdots & \ddots & & \vdots \\ 0 & 0 & 0 & \ldots & s_\ell \\ 1 & 1 & 1 & \ldots & 1 \end{pmatrix} \quad \mathcal{U}_\ell : \mathbf{A} = \begin{pmatrix} s_{1,1} & \ldots & s_{1,\ell} \\ \vdots & & \vdots \\ s_{\ell+1,1} & \ldots & s_{\ell+1,\ell} \end{pmatrix},$$

where $s, s_i, s_{i,j} \leftarrow \mathbb{Z}_p$. Up to sign, the $\mathcal{SC}_\ell$ assumption, introduced in [42], is the *ℓ-symmetric cascade assumption* (ℓ-SCasc). The $\mathcal{L}_\ell$ assumption is actually the ℓ-LIN assumption in matrix language, and the $\mathcal{U}_\ell$ assumption is the *ℓ-uniform assumption*. More generally, we can also define the $\mathcal{U}_{n,\ell}$ assumption for arbitrary $n > \ell$. Note that the $\mathcal{U}_{n,\ell}$ assumption is the weakest matrix assumption (with the worst representation size) and implied by any other $\mathcal{D}_{n,\ell}$ assumption [42]. In particular ℓ-Lin implies the ℓ-uniform assumption as shown by Freeman.

As before, we can switch matrix assumptions to the multilinear setting by using a symmetric multilinear group generator $\mathcal{G}_k$ instead of $\mathcal{G}_1$. If we use a k-linear

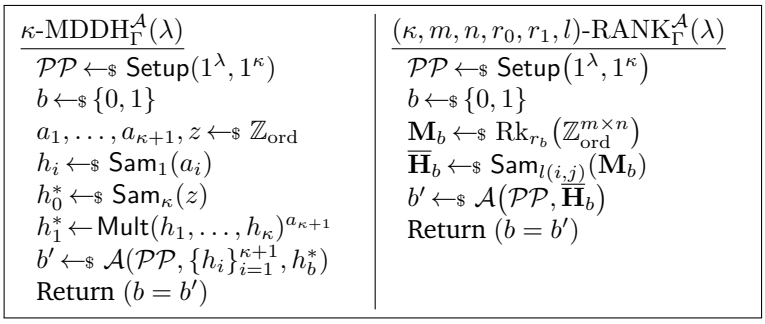

Figure 2.2: Left: The MDDH security game. The sampler algorithms output canonical encodings. The κ-ary algorithm Mult is defined by applying the 2-ary algorithm Mult of the scheme iteratively to inputs. **Right**: The security game for RANK. Here, $\mathrm{Rk}_r(\mathbb{Z}_p^{m\times n})$ denotes the uniform distribution over $m \times n$ matrices of rank r in $\mathbb{Z}_p$. Entries of the matrix $\overline{\mathbf{H}}_b$ encoding $\mathbf{M}_b$ belong to different (nonzero) levels as specified by the function l.

group generator with $k \leq \ell$, [42] shows that the ℓ-SCasc, ℓ-Lin, and the ℓ-uniform assumption hold in the generic group model [110]. On the other hand, every $\mathcal{D}_\ell$ assumption is trivially false using a k-linear group generator with $k > \ell$.

MULTILINEAR ASSUMPTIONS. All assumptions introduced so far share the property that the elements given to the adversary lie in the source group. This will change now that we introduce assumptions where the adversary also receives elements from the target group of a multilinear map. Since, in Chapter 3, we aim at constructing a GES where such multilinear assumptions hold, we will formulate the assumptions using the syntax of Definition 2.3.4 instead of using a multilinear group generator.

THE κ-MDDH PROBLEM [17, 51]. For $\kappa \in \mathbb{N}$ we say that the κ-MDDH problem is hard for a GES $\Gamma := (R, S)$ with $|R| = \mathrm{ord}$ if

$$\mathbf{Adv}^{\kappa\text{-mddh}}_{\Gamma,\mathcal{A}}(\lambda) := 2 \cdot \Pr\left[\kappa\text{-MDDH}^{\mathcal{A}}_{\Gamma}(\lambda)\right] - 1 \in \text{NEGL},$$

where game κ-MDDH$^{\mathcal{A}}_{\Gamma}(\lambda)$ is shown in Figure 2.2 (left).

THE $(\kappa, m, n, r_0, r_1, l)$-RANK PROBLEM [42]. For $\kappa, m, n, r_0, r_1 \in \mathbb{N}$ and a level function $l : [m] \times [n] \longrightarrow [\kappa]$, we say that the $(\kappa, m, n, r_0, r_1, l)$-RANK problem is hard for a GES $\Gamma := (R, S)$ with $|R| = \mathrm{ord}$ if

$$\mathbf{Adv}^{(\kappa,m,n,r_0,r_1,l)\text{-rank}}_{\Gamma,\mathcal{A}}(\lambda) := 2 \cdot \Pr\left[(\kappa, m, n, r_0, r_1, l)\text{-RANK}^{\mathcal{A}}_{\Gamma}(\lambda)\right] - 1 \in \text{NEGL},$$

where game $(\kappa, m, n, r_0, r_1, l)$-RANK$^{\mathcal{A}}_{\Gamma}(\lambda)$ is shown in Figure 2.2 (right).

2.5 Lossy Trapdoor Functions

Definition 2.5.1 (Lossy trapdoor functions ([99])). *Let n, l be polynomial functions in the security parameter λ with $l \leq n$. A collection of (n, l)-*lossy trapdoor functions *(LTDF) consists of the following four PPT algorithms:*

- ***Injective sampling.*** *S_{inj} takes as input n and l and outputs the description of an injective function f over the domain $\{0,1\}^n$ along with a trapdoor td_f.*

- ***Lossy sampling.*** *S_{loss} takes as input n and l and outputs the description of a lossy function f over the domain $\{0,1\}^n$ with image size at most 2^{n-l}.*
- ***Function evaluation.*** *$\mathbb{F}$ takes as input the description of a function f generated by S_{inj} or S_{loss} as well as an input vector $x \in \{0,1\}^n$ and outputs $f(x)$.*
- ***Inversion.*** *F_{ltdf}^{-1} takes as inputs a tuple (f, td_f) generated by S_{inj} and an image $f(x)$ and outputs x.*

We require that injective and lossy functions f (sampled by S_{inj}, resp. S_{loss}) are computationally indistinguishable.

As mentioned in [99], the indistinguishability of lossy and injective functions already implies that the injective functions are hard to invert without knowing the trapdoor td_f. This is due to the fact that an algorithm with non-negligible advantage in inverting the injective function can be used in a straightforward way to distinguish both types of functions with non-negligible probability.

2.6 Public-Key Encryption

The seminal cryptosystem of ElGamal was the first public-key encryption scheme in the discrete-log based setting. It is closely related to the Diffie–Hellman key exchange as follows: to encrypt a message, one creates a (temporary) Diffie–Hellman key pair, computes the shared key with the Diffie–Hellman public key of the receiver and then uses this shared key to blind the message. The ciphertext contains the blinded message together with the temporary public key. The receiver computes the shared Diffie–Hellman key and unblinds the message. A more formal definition of the cryptosystem follows.

Definition 2.6.1 (ElGamal cryptosystem ([49])**.** *The* ElGamal cryptosystem *is defined by the algorithms* Setup*,* KeyGen*,* Enc *and* Dec *as follows.*

- $\mathsf{Setup}(\lambda)$ *runs a group generator $\mathcal{G}_1(\lambda)$ to obtain (G, ord, g).* Setup *outputs the schemes' public parameters $\mathcal{PP} = (\mathrm{ord}, G, g)$.*
- $\mathsf{KeyGen}(\mathcal{PP})$ *chooses $z \in \mathbb{Z}_{\mathrm{ord}}$ randomly and outputs $(pk, sk) := ([z], z)$.*
- $\mathsf{Enc}(\mathcal{PP}, pk, m; r)$*, with $r, m \in \mathbb{Z}_{\mathrm{ord}}$, outputs $c := (c_1, c_2)$ with $(c_1, c_2) := ([r], pk^r[m])$.*
- $\mathsf{Dec}(\mathcal{PP}, sk, c)$ *parses $c =: (c_1, c_2)$ and outputs $m = \log_g\left(\frac{c_2}{c_1^{sk}}\right)$.*

For convenience, we will often omit $\mathcal{PP}$ from the arguments of the algorithms Enc *and* Dec*, and write $\mathsf{Enc}_{pk}(m; r)$ (or simply $\mathsf{Enc}(m; r)$, when the reference to pk is clear) for $\mathsf{Enc}(\mathcal{PP}, pk, m; r)$.*

A straightforward reduction shows that the above variant of the ElGamal cryptosystem is semantically secure if DDH holds in G. Note that this implies that DLOG is hard in G as well and thus decryption only works for "small" messages m. For convenience, we also allow using Enc and Dec with vectors or matrices by computing encryption and decryption component wise, i.e.,

$$\mathsf{Enc}_{\vec{pk}}(\vec{m}; \vec{r}) := (\mathsf{Enc}_{pk_1}(m_1; r_1), \mathsf{Enc}_{pk_2}(m_2; r_2), \dots).$$

$\underline{\text{IND-CPA}^{\mathcal{A}}_{\Pi}(\lambda)}$:
$(sk, pk) \leftarrow\!\!\$\ \mathsf{Gen}(1^\lambda)$
$(m_1, m_1, st) \leftarrow\!\!\$\ \mathcal{A}_1(pk)$
$b \leftarrow\!\!\$\ \{0,1\}$
$c \leftarrow\!\!\$\ \mathsf{Enc}(m, pk)$
$b' \leftarrow\!\!\$\ \mathcal{A}_2(c, st)$
Return $(b = b')$

Figure 2.3: IND-CPA security of a (homomorphic) PKE scheme.

2.6.1 Homomorphic Public-Key Encryption

SYNTAX. A homomorphic public-key encryption (PKE) scheme for a deterministic circuit family $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$ of arity at most $a(\lambda)$ is a tuple of PPT algorithms $\Pi := (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ such that $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is a conventional public-key encryption scheme with message space $\{0,1\}^\lambda$ and Eval is a *deterministic* algorithm that on input a public key pk a circuit $\mathrm{C} \in \mathcal{C}_\lambda$ and ciphertexts $c_1, \ldots, c_n$ with $n \leq a(\lambda)$ outputs a ciphertext c. Without loss of generality, we assume that secret keys of a homomorphic PKE scheme are the random coins used in key generation. This will allow us to check key pairs for validity.

CORRECTNESS AND COMPACTNESS. We require that scheme $\Pi := (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is *perfectly* correct as a PKE scheme; that is, for any $\lambda \in \mathbb{N}$, any $m \in \{0,1\}^\lambda$, any $(sk, pk) \leftarrow\!\!\$\ \mathsf{Gen}(1^\lambda)$, and any $c \leftarrow\!\!\$\ \mathsf{Enc}(m, pk)$ we have that $\mathsf{Dec}(c, sk) = m$. We also require the FHE scheme to be fully compact in the following sense. For any $\lambda \in \mathbb{N}$, any $m_1, \ldots, m_n \in \{0,1\}^\lambda$ with $n \leq a(\lambda)$, any $\mathrm{C} \in \mathcal{C}_\lambda$, any $(sk, pk) \leftarrow\!\!\$\ \mathsf{Gen}(1^\lambda)$ and any $c_i \leftarrow\!\!\$\ \mathsf{Enc}(m_i, pk)$ we have that $\mathsf{Eval}(pk, \mathrm{C}, c_1, \ldots, c_n)$ is in the range of $\mathsf{Enc}(\mathrm{C}(m_1, \ldots, m_n), pk)$.

A *fully* homomorphic encryption (FHE) scheme is a homomorphic PKE that correctly and compactly supports any circuit family containing polynomial-sized circuits of polynomial arity (for any a priori fixed polynomial bounds on the size and arity). In our constructions, full correctness and compactness are used to ensure that the outputs of the addition and multiplications circuits can be iteratively operated on. This in particular means that our GES is "noise-free" in the sense that its correctness is not affected by the number of operations operated on encodings.

We note that although most homomorphic PKE proposals in the literature are not perfectly correct, this property is usually assumed in the literature (cf. [58]). Indeed, it is plausible that perfectly correct homomorphic PKE can be achieved from standard homomorphic PKE constructions by adapting the probability distribution of the noise to a bounded distribution and applying worst-case bounds in all steps.

SECURITY. The IND-CPA security of a homomorphic PKE scheme is defined identically to a standard PKE scheme without reference to the Dec and Eval algorithms. Formally, we require that for any legitimate PPT adversary $\mathcal{A} := (\mathcal{A}_1, \mathcal{A}_2)$,

$$\mathbf{Adv}^{\text{ind-cpa}}_{\Pi,\mathcal{A}}(\lambda) := 2 \cdot \Pr\left[\text{IND-CPA}^{\mathcal{A}}_{\Pi}(\lambda)\right] - 1 \in \text{NEGL},$$

where game $\text{IND-CPA}^{\mathcal{A}}_{\Pi}(\lambda)$ is shown in Figure 2.3. Adversary $\mathcal{A}$ is legitimate if it outputs two messages of equal lengths.

2.7 Dual-Mode NIZK Proof Systems

In our constructions we will rely on special types of "dual-mode" non-interactive zero-knowledge (NIZK) proof systems. These systems have two common reference string (CRS) generation algorithms that produce indistinguishable CRSs in the "binding" and "hiding" modes. They are also perfectly complete in both modes, perfectly sound and extractable in the binding mode, and perfectly witness indistinguishable (WI) and perfectly zero knowledge (ZK) in the hiding mode. The standard prototype for such schemes are the pairing-based Groth–Sahai proofs [63], and using a generic NP reduction to the satisfiability of quadratic equations we can obtain a suitable proof system for any NP language.[2] We formalize the syntax and security of such proof systems next.

SYNTAX. A (group) setup algorithm $\mathbf{G}$ is a PPT Turing machine that on input 1^λ outputs gpk. A ternary relation $\mathbf{R}(gpk, x, w)$ is a deterministic algorithm that outputs 1 for true or 0 for false. A dual-mode extractable non-interactive zero-knowledge (NIZK) proof system Σ *for setup* $\mathbf{G}$ *and relation* $\mathbf{R}$ consists of six algorithms as follows. (1) $\mathsf{BCRS}(gpk)$ on input gpk in the support of $\mathbf{G}$ outputs a (binding) common reference string crs and an extraction trapdoor td_{ext}; (2) $\mathsf{HCRS}(gpk)$ on input gpk in the support of $\mathbf{G}$ outputs a (hiding) common reference string crs and a simulation trapdoor td_{zk}; (3) $\mathsf{Prove}(gpk, crs, x, w)$ on input gpk a first coordinate in the support of $\mathbf{G}$, a common reference string crs, an instance x, and a witness w, outputs a proof π; (4) $\mathsf{Ver}(gpk, crs, x, \pi)$ on input gpk, crs, an instance x, and a proof π, outputs 1 for accept or 0 for reject; (5) $\mathsf{WExt}(td_{ext}, x, \pi)$ on input an extraction trapdoor td_{ext}, an instance x, and a proof π, outputs a witness w; and (6) $\mathsf{Sim}(td_{zk}, x)$ on input the simulation trapdoor td_{zk} and an instance x, outputs a simulated proof π.

We require the extractable dual-mode NIZK Σ for $(\mathbf{G}, \mathbf{R})$ to meet the following requirements.

CRS INDISTINGUISHABILITY. For $gpk \leftarrow\!\!\$\ \mathbf{G}(1^\lambda)$, the two common reference strings generated through $\mathsf{BCRS}(gpk)$ and $\mathsf{HCRS}(gpk)$ are computationally indistinguishable. Formally, we require the advantage of any PPT adversary $\mathcal{A}$ defined below to be negligible.

$$\mathbf{Adv}^{\mathrm{crs}}_{\Sigma,\mathcal{A}}(\lambda) := 2 \cdot \Pr\left[b \leftarrow\!\!\$ \{0,1\}; gpk \leftarrow\!\!\$\ \mathbf{G}(1^\lambda); (crs_0, td_{ext}) \leftarrow\!\!\$\ \mathsf{BCRS}(gpk); (crs_1, td_{zk}) \leftarrow\!\!\$\ \mathsf{HCRS}(gpk); b' \leftarrow\!\!\$\ \mathcal{A}(gpk, crs_b) : b = b'\right] - 1$$

PERFECT COMPLETENESS UNDER BCRS AND HCRS. For any $\lambda \in \mathbb{N}$, any $gpk \leftarrow\!\!\$\ \mathbf{G}(1^\lambda)$, any $(crs, td_{ext}) \leftarrow\!\!\$\ \mathsf{BCRS}(gpk)$, any (x, w) such that $\mathbf{R}(gpk, x, w) = 1$, and any $\pi \leftarrow\!\!\$\ \mathsf{Prove}(gpk, crs, x, w)$, verification always succeeds, i.e., $\mathsf{Ver}(gpk, crs, x, \pi) = 1$. We require this property to also hold for any choice of hiding CRS.

PERFECT SOUNDNESS UNDER BCRS. For any $\lambda \in \mathbb{N}$, any $gpk \leftarrow\!\!\$\ \mathbf{G}(1^\lambda)$, any common reference string $(crs, td_{ext}) \leftarrow\!\!\$\ \mathsf{BCRS}(gpk)$, any x for which $\mathbf{R}(gpk, x, w) = 0$ for all $w \in \{0,1\}^*$, and any $\pi \in \{0,1\}^*$ we have that $\mathsf{Ver}(gpk, crs, x, \pi) = 0$.

[2] We note that extraction in Groth–Sahai proofs does not recover a witness for all types of statements. (Instead, for some types of statements, only g^{w_i} for a witness variable $w_i \in \mathbb{Z}_p$ can be recovered.) Here, however, we will only be interested in witnesses $w = (w_1, \ldots, w_n) \in \{0,1\}^n$ that are bit strings, in which case extraction always recovers w. (Specifically, extraction will recover g^{w_i} for all i, and thus all w_i too.)

PERFECT EXTRACTION UNDER BCRS. For any $\lambda \in \mathbb{N}$, any $(gpk) \leftarrow\!\!\$\ \mathbf{G}(1^\lambda)$, any common reference string $(crs, td_{ext}) \leftarrow\!\!\$\ \mathsf{BCRS}(gpk)$, any (x, π) with $\mathsf{Ver}(gpk, crs, x, \pi) = 1$, and any $w \leftarrow\!\!\$\ \mathsf{WExt}(td_{ext}, x, \pi)$ we have that $\mathbf{R}(gpk, x, w) = 1$.

PERFECT WITNESS INDISTINGUISHABILITY UNDER HCRS. For any $\lambda \in \mathbb{N}$, any $(gpk) \leftarrow\!\!\$\ \mathbf{G}(1^\lambda)$, any $(crs, td_{zk}) \leftarrow\!\!\$\ \mathsf{HCRS}(gpk)$, and any (x, w_b) such that $\mathbf{R}(gpk, x, w_b) = 1$ for $b \in \{0, 1\}$, the two distributions $\pi_b \leftarrow\!\!\$\ \mathsf{Prove}(gpk, crs, x, w_b)$ are identical.

PERFECT ZERO KNOWLEDGE UNDER HCRS. For any $\lambda \in \mathbb{N}$, any $(gpk) \leftarrow\!\!\$\ \mathbf{G}(1^\lambda)$, any $(crs, td_{zk}) \leftarrow\!\!\$\ \mathsf{HCRS}(gpk)$, any (x, w) such that $\mathbf{R}(gpk, x, w) = 1$, we have that the two distributions $\pi_0 \leftarrow\!\!\$\ \mathsf{Prove}(gpk, crs, x, w)$ and $\pi_1 \leftarrow\!\!\$\ \mathsf{Sim}(td_{zk}, x)$ are identical.

Chapter 3

Graded Encoding Schemes from Obfuscation

In this chapter, we contribute to the question whether there exist a multilinear map or graded encoding scheme, respectively. Namely, we give a construction of a graded encoding scheme. Since we adopt techniques from [3], we also give details of their construction and highlight the differences to our GES in the further course.

3.1 Overview

OUR CONSTRUCTION. Technically, we extend the non-graded multilinear map construction from AFHLP [3] to a GES. Our construction requires several ingredients (see below), but all of them can be instantiated with a pairing-friendly group in which (a suitable variant of) the strong decisional Diffie–Hellman (SDDH) assumption [9] holds, and a sub-exponentially secure indistinguishability obfuscator. We prove that the multilinear decisional Diffie–Hellman (MDDH) assumption [51] holds relative to our GES, provided that the used ingredients are secure in the above sense. Similarly, we show that a variety of "matrix rank problems" [42] holds in our setting, a problem that is known to succumb already to very weak forms of zeroizing attacks [51, Section 4.4].

Our definition of a GES essentially implements the "dream version" of GESs [51], but differs in two aspects:

- GGH do not permit sampling for specific values $a \in R$. (Instead, GGH provide an algorithm to sample a random a along with its encoding.)
- GGH's zero-testing algorithm is substituted with an equality test (through Eq_i) above. Our equality test must only work for *consistent* encodings from some $S_i^{(a)}$ and $S_i^{(b)}$. In contrast, the dream version of GGH requires that the set $S_i^{(0)}$ is efficiently recognizable.

RELATIONS TO INDISTINGUISHABILITY OBFUSCATION. Our construction is generic and modular. In particular, we reduce the quest to develop a secure GES to the quest for a secure indistinguishability obfuscator. This seems natural (and is standard in most areas of cryptography), but given the history of previous GES candidates (which were based on complex algebraic or combinatorial assumptions), this is not an "understood feature" at all for GESs. In fact, taken together with [52], our result shows a (somewhat loose) equivalence of indistinguishability obfuscation ($i\mathcal{O}$) and GESs, in the presence of a pairing-friendly group. This equivalence is loose for two reasons. First, the assumptions on both ends of the equivalence do not match: [52] construct $i\mathcal{O}$ from a GES which support very strong computational assumptions (much stronger than MDDH). On the other hand, we use $i\mathcal{O}$ to construct a GES in which we can (at this point) only prove comparatively mild (though still

useful) computational assumptions (such as MDDH). Still, there seems no inherent barrier to proving stronger computational assumptions for our construction, and we leave open to tighten this equivalence. Second, going through our equivalence is not without (sub-exponential) security loss. Namely, we require *probabilistic* indistinguishability obfuscation, which can be constructed from $i\mathcal{O}$ [28], but currently only through a sub-exponential reduction.

3.1.1 The (Non-Graded) Approximate Multilinear Map of AFHLP

ENCODINGS. Since our own construction is an extension of the (non-graded) approximate multilinear map of [3], we first recall their work. Simplifying slightly, AFHLP encode a group element g^z (from a cyclic group $\mathbb{G}$ of order p) as

$$h \;=\; (g^z, c = \mathsf{Enc}((\alpha,\beta), pk), \pi) \;,$$

where

- c is a homomorphic encryption (under some public key pk) of exponents $\alpha, \beta \in \mathbb{Z}_p$,
- π is a non-interactive zero-knowledge proof that these exponents represent z in the sense that $g^z = g^\alpha u^\beta$ for a publicly known group element u. (Hence, if we write $u = g^\omega$, we have $z = \alpha + \beta \cdot \omega$.)

Hence, AFHLP simply enhance the group element $g^z \in \mathbb{G}$ by an encrypted representation of its discrete logarithm z (and a suitable consistency proof). This added information will be instrumental in computing a multilinear map on many encodings. Note that since c and π will not be uniquely determined, there are many possible encodings of a $\mathbb{G}$-element g^z.

ADDITION. Encodings in the AFHLP construction can be added with an (obfuscated) public circuit Add. This circuit takes as input two encodings $h_1 = (g^{z_1}, c_1, \pi_1)$ and $h_2 = (g^{z_2}, c_2, \pi_2)$, and computes the new encoding $h_1 + h_2 = (g^z, c, \pi)$ as follows:

1. $g^z = g^{z_1+z_2}$ is computed using the group operation in $\mathbb{G}$;
2. c is computed homomorphically from c_1 and c_2 (adding the encrypted exponent vectors (α_i, β_i));
3. the consistency proof π is computed using the decryption key sk as a witness to show that the resulting c indeed contains a valid representation of $z = z_1 + z_2$.

Here, only the computation of π requires secret information (namely, the decryption key sk). This secret information allows to derive a valid representation (α, β) of g^z. The most delicate part of the security proof from [3] is to argue that the obfuscated circuit knowing sk does not help in solving (a multilinear variant of) the decisional Diffie–Hellman problem.

THE MULTILINEAR MAP. The AFHLP encodings can also be multiplied with an (obfuscated) public circuit Mult; this takes as input κ encodings $h_1, \ldots, h_\kappa$ with $h_i = (g^{z_i}, c_i, \pi_i)$, and outputs a single group element $g^{\prod_{i=1}^{\kappa} z_i}$. (Hence, elements from the target group $\mathbb{G}_T$ are trivially and uniquely encoded as $\mathbb{G}$-elements.) To compute $g^{\prod z_i}$ from the h_i, Mult first checks the validity of all proofs π_i, and then uses the decryption key sk to retrieve representations (α_i, β_i). If all π_i are verifying proofs,

we may assume that $z_i = \alpha_i + \beta_i \cdot \omega$ (for $u = g^\omega$), so we can write

$$g^{\prod_{i=1}^{\kappa} z_i} \;=\; \prod_{i=0}^{\kappa} (g^{\omega^i})^{\gamma_i} \quad \text{for} \quad (\gamma_0, \ldots, \gamma_\kappa) = (\alpha_1, \beta_1) * \cdots * (\alpha_\kappa, \beta_\kappa)\ , \tag{3.1}$$

where "$*$" denotes the convolution product of vectors.[1] The values g^{ω^i} (for $i \leq \kappa$) are hardwired into Mult, so Mult can compute $g^{\prod z_i}$ through Equation (3.1). Note that this way, Mult can compute a κ-linear map on encodings, but not a $(\kappa+1)$-linear map. This observation is the key to showing that the MDDH assumption holds in this setting. (Indeed, the MDDH assumption states that given $\kappa + 1$ encodings $h_1, \ldots, h_{\kappa+1}$ as above, it is hard to distinguish $g^{\prod_{i=1}^{\kappa+1} z_i}$ from random.)

3.1.2 Our New Graded Encoding Scheme

In the following, we will describe the main ideas for our GES.

ENCODINGS IN OUR SCHEME. In our GES, we generalize the linear representation of exponents in AFHLP to polynomials of higher degree. Additionally, we divide encodings into levels by restricting the maximum degree of the representing polynomial in each level. More formally, level-ℓ encodings take the form

$$h \;=\; (g^z, c = \mathsf{Enc}(P, pk), \pi, \ell)\ ,$$

where

- $g^z \in \mathbb{G}$ for a cyclic group $\mathbb{G}$ (that does not depend on ℓ) of prime order p,
- $P \in \mathbb{Z}_p[X]$ is a polynomial of degree up to ℓ, represented by its coefficient vector from $\mathbb{Z}_p^{\ell+1}$,
- c is the encryption (under a fully homomorphic encryption scheme) of P,
- π is a non-interactive zero-knowledge proof of the equality $g^z = g^{P(\omega)}$, where ω is defined through public values $u_0, \ldots, u_\kappa \in \mathbb{G}$ with $u_i = g^{\omega^i}$. (Hence, $g^z = g^{P(\omega)}$ is equivalent to $g^z = \prod_i u_i^{\gamma_i}$ for $P(X) = \sum_i \gamma_i X^i$.)

The encodings of AFHLP can be viewed as level-1 encodings in our scheme (with linear polynomials P).

ADDING ENCODINGS. Encodings can be added using a public (obfuscated) circuit Add that proceeds similarly to the AFHLP scheme. In particular, Add adds the g^z and c parts of the input encodings homomorphically, and derives a consistency proof π with the decryption key sk as witness.

MULTIPLYING ENCODINGS. The pairings $e_{i,j} : \mathbb{G}_i \times \mathbb{G}_j \longrightarrow \mathbb{G}_{i+j}$ are implemented over our encodings by (obfuscated) circuits $\mathsf{Mult}_{i,j}$. Circuit $\mathsf{Mult}_{i,j}$ takes as input two encodings $h_1 = (g^{z_1}, c_1, \pi_1, i)$ and $h_2 = (g^{z_2}, c_2, \pi_2, j)$ at levels i and j, respectively. The output of $\mathsf{Mult}_{i,j}$ is a level-$(i + j)$ encoding $h = (g^z, c, \pi, i + j)$, computed as follows:[2]

[1] Recall that the multiplication of polynomials can be implemented through the convolution product on the respective coefficient vectors. In particular, we have $\prod_{i=0}^{\kappa} \gamma_i X^i = \prod_{i=1}^{\kappa} (\alpha_i + \beta_i X)$.

[2] Since $\mathsf{Mult}_{i,j}$ can be used to multiply two encodings at level i as long as $2i \leq \kappa$, our GES can be viewed as *symmetric*. We note that we do not deal with the construction of generalized GES.

- g^z is computed as $g^z = g^{(P_1 \cdot P_2)(\omega)}$, where the polynomials P_1 and P_2 are extracted from c_1 and c_2 with sk, then multiplied to form $P := P_1 \cdot P_2 \in \mathbb{Z}_p[X]$, and finally used to compute

$$g^{(P_1 \cdot P_2)(\omega)} \;=\; g^{P(\omega)} \;=\; \prod_{\ell=0}^{i+j} u_\ell^{\gamma_\ell} \quad \text{for} \quad P(X) = \sum_{\ell=0}^{i+j} \gamma_\ell X^\ell \,.$$

  (Note that since $u_0, \ldots, u_\kappa$ are public, this value can be computed as long as $i + j \leq \kappa$.)
- c is computed homomorphically from c_1 and c_2, as an encryption of the polynomial $P_1 \cdot P_2$.
- The consistency proof π (showing that indeed $g^z = g^{P(\omega)}$ for the polynomial P encrypted in c) is computed with the decryption key sk as witness.

The key insight needed to show that the MDDH assumption holds for our GES is the same as in AFHLP's non-graded, approximate multilinear map. Namely, observe that any $\mathsf{Mult}_{i,j}$ can only multiply encodings if $i + j \leq \kappa$. To compute the first component g^z of any "higher-level" encoding, one would seem to require g^{ω^ℓ} values for $\ell > i + j$. Under the SDDH assumption in $\mathbb{G}$, such g^{ω^ℓ} look random, even when given $u_0, \ldots, u_\kappa$. Of course, to turn this observation into a full proof, more work is required.

NEGLECTED DETAILS. For a useful GES, it should be possible to generate encodings with "known discrete logarithm"; that is, we would like to be able to generate encodings for an externally given (or at least known) $z \in \mathbb{Z}_p$. For this reason, the standard way to generate encodings (at any level) is to set up P as a *constant* polynomial of the form $P(X) = z \in \mathbb{Z}_p$. (That is, we "reserve space" in c for polynomials P of degree ℓ in level-ℓ encodings, but, by default, use only constant polynomials.) For this type of encoding with "low-degree P," however, our security argument above does not apply. Rather, it requires that the degree of P increases at higher levels.

Hence, the central technical piece in our MDDH security proof will be a "switching theorem" that allows to replace a low-degree P in an encoding with an *equivalent* high-degree P' (that satisfies $P'(\omega) = P(\omega)$). The proof of this switching theorem is delicate, since it must work in a setting with (obfuscated) algorithms that use the decryption key sk. (Note that free access to sk would allow the retrieval of the used polynomial P from an encoding, and hence would prevent such a switching of polynomials.)

To this end, we will use *double encryptions* c (instead of the single encryption $c = \mathsf{Enc}(P, pk)$ described above), along with a Naor–Yung-style consistency proof in π. However, this consistency proof does not show equality of encryptions, but *equivalence* of encrypted representations P, P' in the sense of $P(\omega) = P'(\omega)$. This allows to switch representations without invalidating the consistency of the double encryption. As a result, the full consistency language used for π is considerably more complicated than the one sketched before. Additionally, the proof of our switching theorem requires a special and explicit "simulation trapdoor" and Groth–Sahai-style dual-mode proof systems.

We note that similar complications arose already in AFHLP's proof, and required similar measures. The main technical difference in our setting is that our multiplication circuits $\mathsf{Mult}_{i,j}$ output *encodings* (and not just group elements as in the

multilinear map of AFHLP). Hence, our $\mathsf{Mult}_{i,j}$ circuits also need to construct consistency proofs π, which requires additional secrets (as witnesses) in the description of $\mathsf{Mult}_{i,j}$ and which entails additional steps in our switching theorem. (We give more details on the technical differences with AFHLP in the main body. However, we note that, in addition to providing a *graded* encoding scheme, we also provide simplified proofs and a single construction in which *both* the MDDH and a "matrix rank" assumption, which we call RANK, hold simultaneously.)

ASSUMPTIONS. In summary, our construction uses a cyclic group in which the SDDH assumption holds, a probabilistic indistinguishability obfuscation scheme [28], a fully homomorphic encryption (FHE), a dual-mode non-interactive zero-knowledge proof systems, and a language with hard membership. However, we note that all of these assumptions are implied by pairing-friendly SDDH groups (equipped with an asymmetric pairing) and sub-exponentially secure indistinguishability obfuscation (see [64, 28]). We stress that plausible candidates for both ingredients exist (e.g., by combining [51] and [52] to an indistinguishability obfuscator candidate).

3.2 Preliminaries

CIRCUITS. A polynomial-sized deterministic circuit family $\mathcal{C} := \{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$ is a sequence of sets $\mathcal{C}_\lambda$ of $\mathrm{poly}(\lambda)$-sized deterministic circuits (for a fixed polynomial $\mathrm{poly}(\lambda)$). We assume that for all $\lambda \in \mathbb{N}$ all circuits $\mathrm{C} \in \mathcal{C}_\lambda$ share a common input domain $(\{0,1\}^\lambda)^{a(\lambda)}$, where $a(\lambda)$ is the arity of the circuit family, and an output co-domain $\{0,1\}^\lambda$. A randomized circuit family is defined similarly except that the circuits also take random coins $r \in \{0,1\}^{\mathrm{rl}(\lambda)}$, for a polynomial $\mathrm{rl}(\lambda)$ specifying the length of necessary random coins. To make the coins used by a circuit explicit (e.g., to view a randomized circuit as a deterministic one) we write $\mathrm{C}(x;r)$.

3.2.1 Languages with Hard Membership

In our proofs of security we also rely on languages for which the membership problem is hard and whose yes-instances have unique witnesses. Formally, such a language family is defined as a triple of PPT algorithms $\Lambda := (\mathsf{Gen}_\mathsf{L}, \mathsf{YesSam}_\mathsf{L}, \mathsf{NoSam}_\mathsf{L}, \mathsf{R}_\mathsf{L})$ as follows. (1) $\mathsf{Gen}_\mathsf{L}(1^\lambda)$ is randomized and on input the security parameter outputs a language key lk; (2) $\mathsf{YesSam}_\mathsf{L}(lk)$ is randomized and on input the language key lk outputs a yes-instance y; (3) $\mathsf{NoSam}_\mathsf{L}(lk)$ is randomized and on input the language key lk outputs a no-instance y; and (4) $\mathsf{R}_\mathsf{L}(lk, y, w)$ is deterministic and on input lk, an instance y and a witness w outputs 1 for true or 0 for false.

We require R_L to satisfy the following correctness requirements. For all $\lambda \in \mathbb{N}$, all $lk \leftarrow\!\!\$\ \mathsf{Gen}_\mathsf{L}(1^\lambda)$ and all $y \leftarrow\!\!\$\ \mathsf{YesSam}_\mathsf{L}(lk)$ there is a $w \in \{0,1\}^*$ such that $\mathsf{R}_\mathsf{L}(lk, y, w) = 1$. For a given lk, we denote the set of yes-instance by $\mathcal{L}_{lk}$. For all $\lambda \in \mathbb{N}$, all $lk \leftarrow\!\!\$\ \mathsf{Gen}_\mathsf{L}(1^\lambda)$ and all $y \leftarrow\!\!\$\ \mathsf{NoSam}_\mathsf{L}(lk)$ there is no $w \in \{0,1\}^*$ such that $\mathsf{R}_\mathsf{L}(lk, y, w) = 1$. We also require R_L to have unique witnesses: for all $\lambda \in \mathbb{N}$, all $lk \leftarrow\!\!\$\ \mathsf{Gen}_\mathsf{L}(1^\lambda)$, all $y \leftarrow\!\!\$\ \mathsf{YesSam}_\mathsf{L}(lk)$ and all $w, w' \in \{0,1\}^*$ if $\mathsf{R}_\mathsf{L}(lk, y, w) = \mathsf{R}_\mathsf{L}(lk, y, w') = 1$ then $w = w'$.

Finally, the language is required to have a hard membership problem in the sense

that for any PPT adversary $\mathcal{A}$

$$\mathbf{Adv}^{\mathrm{mem}}_{\Lambda,\mathcal{A}}(\lambda) := 2 \cdot \Pr\left[b \leftarrow\!\!\$\, \{0,1\}; lk \leftarrow\!\!\$\, \mathsf{Gen}_{\mathsf{L}}(1^\lambda); y_0 \leftarrow\!\!\$\, \mathsf{NoSam}_{\mathsf{L}}(lk); \right.$$
$$\left. y_1 \leftarrow\!\!\$\, \mathsf{YesSam}_{\mathsf{L}}(lk); b' \leftarrow\!\!\$\, \mathcal{A}(lk, y_b) : b = b'\right] - 1 \in \textsc{Negl}\,.$$

Such languages can be instantiated using the DDH problem as follows. Algorithm $\mathsf{Gen}_{\mathsf{L}}(1^\lambda)$ outputs the description of a prime-order group $(\mathbb{G}, g, p, 1)$ as lk. Algorithm $\mathsf{YesSam}_{\mathsf{L}}(lk)$ samples a Diffie–Hellman tuple (g^a, g^b, g^{ab}), and $\mathsf{NoSam}_{\mathsf{L}}(lk)$ outputs a non-Diffie–Hellman tuple (g^a, g^b, g^c) for a random $c \neq ab \pmod p$ when $b = 0$. Relation R_{L} on instance (g_1, g_2, g_3) and witness $w = a$ checks if $g_1 = g^a$ and $g_3 = g_2^a$. The hardness of membership for this language family follows from the DDH assumption.

3.2.2 Obfuscators

SYNTAX AND CORRECTNESS. A PPT algorithm Obf is called an *obfuscator* for a (deterministic or randomized) circuit class $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$ if Obf on input the security parameter 1^λ and the description of a (deterministic or randomized) circuit $\mathrm{C} \in \mathcal{C}_\lambda$ of arity $a(\lambda)$ outputs a *deterministic* circuit $\overline{\mathrm{C}}$. For deterministic circuits, we require Obf to be perfectly correct in the sense the circuits C and $\overline{\mathrm{C}}$ are functionally equivalent; that is, that for all $\lambda \in \mathbb{N}$, all $\mathrm{C} \in \mathcal{C}_\lambda$, all $\overline{\mathrm{C}} \leftarrow\!\!\$\, \mathsf{Obf}(1^\lambda, \mathrm{C})$, and all $m_i \in \{0,1\}^\lambda$ for $i \in [a(\lambda)]$ we have that $\mathrm{C}(m_1, \ldots, m_{a(\lambda)}) = \overline{\mathrm{C}}(m_1, \ldots, m_{a(\lambda)})$. For randomized circuits, the authors of [28] define correctness via computational indistinguishability of the outputs of C and $\overline{\mathrm{C}}$. For our constructions we do *not* rely on this property and instead require that C and $\overline{\mathrm{C}}$ are functionally equivalent up to a change in randomness; that is, for all $\lambda \in \mathbb{N}$, all $\mathrm{C} \in \mathcal{C}_\lambda$, all $\overline{\mathrm{C}} \leftarrow\!\!\$\, \mathsf{Obf}(1^\lambda, \mathrm{C})$ and all $m_i \in \{0,1\}^\lambda$ for $i \in [a(\lambda)]$ there is an r such that $\overline{\mathrm{C}}(m_1, \ldots, m_{a(\lambda)}) = \mathrm{C}(m_1, \ldots, m_{a(\lambda)}; r)$.We note that the construction from [28] is correct in this sense as it relies on a correct indistinguishability obfuscator and a PRF to internally generate the required random coins.

SECURITY. The security of an obfuscator Obf requires that for any legitimate PPT adversary $\mathcal{A} := (\mathcal{A}_1, \mathcal{A}_2)$

$$\mathbf{Adv}_{\mathsf{Obf},\mathcal{A}}(\lambda) := 2 \cdot \Pr\left[\mathrm{IND}^{\mathcal{A}}_{\mathsf{Obf}}(\lambda)\right] - 1 \in \textsc{Negl},$$

where game IND is shown in Figure 3.1 (left). Depending on the adopted notion of legitimacy, different security notions for the obfuscator emerge; we consider the following one.

X-IND SAMPLERS [28]. Roughly speaking, the first phase of $\mathcal{A} := (\mathcal{A}_1, \mathcal{A}_2)$ is an X-IND sampler if there is a set $\mathcal{X}$ of size at most X such that the circuits output by $\mathcal{A}$ are functionally equivalent outside $\mathcal{X}$, and furthermore within $\mathcal{X}$ the outputs of the circuits are computationally indistinguishable. Formally, let $X(\cdot)$ be a function such that $X(\lambda) \leq 2^\lambda$ for all $\lambda \in \mathbb{N}$. We call $\mathcal{A} := (\mathcal{A}_1, \mathcal{A}_2)$ an X-IND *sampler* if there are sets $\mathcal{X}_\lambda$ of size at most $X(\lambda)$ such that the following two conditions holds:

(1) For all (even unbounded) $\mathcal{D}$ the advantage function below is negligible.

$$\mathbf{Adv}^{\mathrm{eq}}_{\mathcal{A},\mathcal{D}}(\lambda) := \Pr\left[(\mathrm{C}_0, \mathrm{C}_1, st) \leftarrow\!\!\$\, \mathcal{A}_1(1^\lambda); (x, r) \leftarrow\!\!\$\, \mathcal{D}(\mathrm{C}_0, \mathrm{C}_1, st) : \right.$$
$$\left. \mathrm{C}_0(x; r) \neq \mathrm{C}_1(x; r) \wedge x \notin \mathcal{X}_\lambda\right]$$

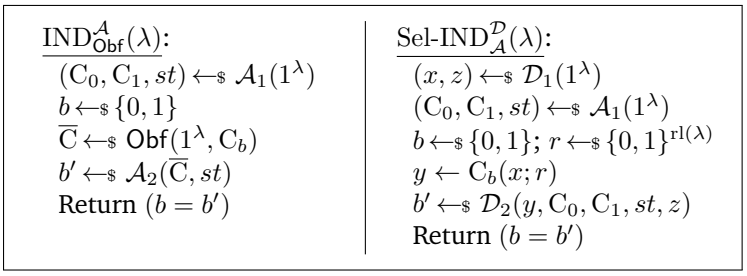

Figure 3.1: Left: Indistinguishability security of an obfuscator. We require $\mathcal{A}_1$ to output two circuits of equal sizes. **Right**: Static-input (a.k.a. selective) X-IND property of $\mathcal{A} := (\mathcal{A}_1, \mathcal{A}_2)$.

(2) For all non-uniform PPT distinguishers $\mathcal{D} := (\mathcal{D}_1, \mathcal{D}_2)$

$$X(\lambda) \cdot \mathbf{Adv}^{\text{sel-ind}}_{\mathcal{A},\mathcal{D}}(\lambda) := X(\lambda) \cdot \left(2 \Pr\left[\text{Sel-IND}^{\mathcal{D}}_{\mathcal{A}}(\lambda)\right] - 1\right) \in \text{NEGL},$$

where game $\text{Sel-IND}^{\mathcal{D}}_{\mathcal{A}}(\lambda)$ is shown in Figure 3.1 (right). This game is named "static-input-IND" in [28] and has a selective (or static) flavor since $\mathcal{D}_1$ chooses a differing-input x *before* it gets to see the challenge circuits. We call an obfuscator meeting this level of security a *probabilistic indistinguishability obfuscator* [28] and use **PIO** instead of Obf to emphasize this.

REMARK. We note that samplers that output two (possibly randomized) circuits $(\mathrm{C}_0, \mathrm{C}_1)$ for which the output distributions of $\mathrm{C}_0(x)$ and $\mathrm{C}_1(x)$ are identical on any input x, are Sel-IND-secure for any function $X(\lambda)$. The circuits samplers that we will use in our security proofs enjoy this property.

3.3 Approximate Multilinear Maps

In this section we recall the approximate multilinear maps due to AFHLP [3]. The authors construct both symmetric and asymmetric multilinear maps. Their symmetric construction can be seen as a starting point for our GES, and we give an overview of it here.

3.3.1 Syntax

We start with the syntax of multilinear group (MLG) schemes [3]. Informally, a κ-MLG scheme is a restricted form of a graded encoding scheme where encodings belong to levels 0, 1 and κ only and the Mult algorithm takes κ encodings at level 1 and outputs an encoding at level κ. We formalize MLG schemes using the notion introduced for GESs below.

SYMMETRIC MLG SCHEMES. A symmetric κ-linear group scheme is a κ-graded encoding scheme associated to (R, S), where (R, S) is defined similarly to a κ-graded encoding system except that $S := \{S_i^{(a)} \subset \{0,1\}^* \ : \ a \in R, i \in \{0, 1, \kappa\}\}$ and the "$\times$" operation is redefined as a κ-ary map that for any $a_1, \ldots, a_\kappa \in R$ and any $u_1 \in S_1^{(a_1)}, \ldots, u_\kappa \in S_1^{(a_\kappa)}$ satisfies

$$u_1 \times \cdots \times u_\kappa \in S_\kappa^{(a_1 \cdots a_\kappa)}.$$

The associated Mult algorithm on inputs $h_i \in S_1^{(a_i)}$ for $i \in [\kappa]$ outputs an encoding in $S_\kappa^{(a_1 \cdots a_\kappa)}$. Algorithms Eq, Add, Sam and Ext are defined analogously and restricted to $i \in \{0, 1, \kappa\}$ only.

3.3.2 Overview of AFHLP

In a nutshell, [3] works with redundant encodings of elements h of the base group $\mathbb{G}$ of the form $h = g^{x_0}(g^\omega)^{x_1}$ where g^ω comes from an SDDH instance. Vector $\vec{x} = (x_0, x_1)$ *represents* element h. The set S_1 consists of all strings of the form (h, c_1, c_2, π) where $h \in \mathbb{G}$, ciphertext c_1 is a homomorphic encryption under public key pk_1 of a vector $\vec{x}$ representing h, ciphertext c_2 is a homomorphic encryption under a second public key pk_2 of another vector $\vec{y}$ also representing h, and π is a NIZK proof showing consistency of the two vectors $\vec{x}$ and $\vec{y}$. Here consistency means that the plaintexts vectors $\vec{x}$ and $\vec{y}$ underlying c_1 and c_2 encode the same group element h. Note that each element of the base group $\mathbb{G}$ is multiply represented in S_1, but that equality of elements in S_1 is easy to test (via checking the equality of first components).

Addition of two elements in S_1 is carried out by an obfuscation of a circuit $\mathrm{C}_{\mathrm{Add}}[sk_1, sk_2]$, which has the two secret keys hardwired in, as follows. The circuit checks the respective proofs, adds the group elements in $\mathbb{G}$ and uses the additive homomorphic property of the encryption scheme to combine ciphertexts. It then uses (sk_1, sk_2) as a witness to generate a new NIZK proof showing the equality of encodings. Note that the new encoding is as compact as those for the two input encodings.

The multilinear map on inputs $(h_i, c_{i,1}, c_{i,2}, \pi_i)$ for $1 \le i \le \kappa$ is computed using an obfuscation of a circuit $\mathrm{C}_{\mathrm{Map}}[sk_1, \omega]$, which has sk_1 and ω hardwired in, as follows. The circuit recovers the exponents of h_i in the form $(x_{i,1} + \omega \cdot x_{i,2})$ from $c_{i,1}$ via the decryption algorithm $\mathsf{Dec}(\cdot, sk_1)$. It then uses these to compute the group element $g^{\prod_i (x_{i,1} + \omega \cdot x_{i,2})}$, which is defined to be the output of Mult. (The target set S_κ is therefore $\mathbb{G}$, the base group.) The κ-linearity of Mult follows immediately from the form of the exponent.

In the original paper, this construction is generalized to the asymmetric setting via representations of the form $g^{\langle \vec{x}, \vec{\omega} \rangle}$ with $\vec{x}, \vec{\omega} \in \mathbb{Z}_N^\ell$ for $\ell \in \{2, 3\}$ (where $\langle \vec{x}, \vec{\omega} \rangle$ denotes inner products modulo the base-group order). Two special cases of $\vec{\omega}$, namely $\vec{\omega} := (1, \omega)$ and $\vec{\omega} := (1, \omega, \omega^2)$, are used to construct two MLG schemes where the MDDH and RANK problems are hard, respectively.

In more detail, AFHLP [3] construct a symmetric κ-linear group scheme Γ relying on the following building blocks:

1. An algorithm $\mathsf{Setup}_{\mathbb{G}}$ that samples (a description of) a group $\mathbb{G}$, along with a generator g of $\mathbb{G}$ and the group order p.
2. A probabilistic indistinguishability obfuscator Obf.
3. An *additively* homomorphic public-key encryption scheme Π with plaintext space $\mathbb{Z}_p$ (or alternatively, a perfectly correct FHE scheme).
4. An extractable dual-mode NIZK proof system Σ.
5. A language family Λ with hard membership problem and unique witnesses.

We recall their construction in the sections that follow.

SETUP. The algorithm Setup for the GES Γ gets as input 1^λ and 1^κ. It samples

parameters $\mathcal{PP}_{\mathbb{G}} \leftarrow\!\!\$\ \mathsf{Setup}_{\mathbb{G}}(1^\lambda)$ with $\mathcal{PP}_{\mathbb{G}} := (\mathbb{G}, g, p, 1)$, generates two encryption key pairs $(pk_j, sk_j) \leftarrow\!\!\$\ \mathsf{Gen}(1^\lambda)$ (for $j = 1, 2$), and a vector $\vec{\omega} \in \mathbb{Z}_p^\ell$ where $\ell \in \{2, 3\}$. $\mathbb{G}$ is called the *base group*. It then samples $lk \leftarrow\!\!\$\ \mathsf{Gen}_{\mathsf{L}}(1^\lambda)$, and sets

$$gpk := (\mathcal{PP}_{\mathbb{G}}, pk_1, pk_2, [\vec{\omega}], lk).$$

Let $\mathbf{G}(1^\lambda)$ denote the randomized algorithm corresponding to the above steps that outputs gpk.

The setup algorithm continues by generating a CRS $crs' \leftarrow\!\!\$\ \mathsf{BCRS}(gpk)$ using the dual-mode NIZK procedure BCRS, and also a no-instance of $\mathcal{L}_{lk}$ via $y \leftarrow\!\!\$\ \mathsf{NoSam}_{\mathsf{L}}(lk)$. Setup then sets $crs := (crs', y)$.

Finally, Setup constructs two obfuscated circuits $\overline{\mathrm{C}}_{\mathrm{Map}}$ and $\overline{\mathrm{C}}_{\mathrm{Add}}$ of circuits $\mathrm{C}_{\mathrm{Map}}$ and $\mathrm{C}_{\mathrm{Add}}$ which will be described in the addition and multilinear map routines below, respectively. Setup then outputs the scheme parameters

$$\mathcal{PP} := (gpk, crs, \overline{\mathrm{C}}_{\mathrm{Add}}, \overline{\mathrm{C}}_{\mathrm{Map}}).$$

LEVEL-0 ENCODINGS. The set of all level-0 encodings, S_0, is defined to be $\mathbb{Z}_p$. Since efficient algorithms for equality checking, sampling, extraction and addition are well known, we omit including these in the following sections. Note that the algorithm for adding encodings, which is described below, can be used to implement a multiplication of level-0 encodings with encodings at higher levels, which is required by many applications.

LEVEL-κ ENCODINGS. Set $S_\kappa := \mathbb{G}$ and use algorithms associated with $\mathbb{G}$ for equality checking, sampling, extraction and addition.

LEVEL-1 ENCODINGS. Encodings in S_1 are tuples of the form $h = ([z], c_1, c_2, \pi)$ where c_1, c_2 are two ciphertext in the range of $\mathsf{Enc}(\cdot, pk_1)$ and $\mathsf{Enc}(\cdot, pk_2)$, respectively, and π is a NIZK proof under crs for a proof system corresponding to $(\mathbf{G}, \mathbf{R} := \mathbf{R}_1 \vee \mathbf{R}_2)$ as follows. Algorithm $\mathbf{G}(1^\lambda)$ outputs gpk as defined above. Relation $\mathbf{R}_1$ on input gpk, tuple $([z], c_1, c_2)$, and witness $(\vec{x}, \vec{y}, \vec{r}_1, \vec{r}_2, sk_1, sk_2)$ accepts iff $[z] \in \mathbb{G}$, the *representations* of $[z]$ as $\vec{x}, \vec{y} \in \mathbb{Z}_p^\ell$ are valid with respect to $[\vec{\omega}]$ in the sense that

$$[z] = [\langle \vec{x}, \vec{\omega} \rangle] \wedge\ [z] = [\langle \vec{y}, \vec{\omega} \rangle],$$

(where $\langle \cdot, \cdot \rangle$ denotes inner product) and the following ciphertext validity condition (with respect to the inputs to the relation) is met:

$$c_1 = \mathsf{Enc}(\vec{x}, pk_1; \vec{r}_1) \wedge c_2 = \mathsf{Enc}(\vec{x}, pk_2; \vec{r}_2)$$
$$\vee$$
$$(pk_1, sk_1) = \mathsf{Gen}(sk_1) \wedge (pk_2, sk_2) = \mathsf{Gen}(sk_2) \wedge \vec{x} = \mathsf{Dec}(c_1, sk_1) \wedge \vec{y} = \mathsf{Dec}(c_2, sk_2).$$

Relation $\mathbf{R}_2$ depends on Λ and on input gpk, an encoding $([z], c_1, c_2)$, and witness w_y accepts iff $\mathbf{R}(lk, y, w_y)$ accepts. We note that AFHLP does *not* come with a validity check for encodings for the same reason our construction fails to provide such an algorithm. (See Section 3.4.2 for more details.)

EQUALITY. The equality algorithm Eq_1 returns true iff their first components match in $\mathbb{G}$. The correctness follows from the fact that $\mathbb{G}$ has unique encodings.

Circuit $\mathrm{C}_{\mathrm{Add}}[gpk, crs, sk_1, sk_2, td_{ext}; \xleftarrow{R}](h, h')$:
1. if $\neg\mathsf{Val}_1(h) \vee \neg\mathsf{Val}_1(h')$ return $\perp$
2. parse $([z], c_1, c_2, \pi) \leftarrow h$ and $([z'], c_1', c_2', \pi') \leftarrow h'$
3. $[z''] \leftarrow [z] + [z']$; $c_1'' \leftarrow c_1 + c_1'$; $c_2'' \leftarrow c_2 + c_2'$
4. // explicitly check relation $\mathbf{R}_1$ for h, h' with witness sk_1, sk_2
   4.1 $\vec{x} \leftarrow \mathsf{Dec}(c_1, sk_1); \vec{y} \leftarrow \mathsf{Dec}(c_2, sk_2)$
   $\vec{x}' \leftarrow \mathsf{Dec}(c_1', sk_1); \vec{y}' \leftarrow \mathsf{Dec}(c_2', sk_2)$
   4.2a if $([z] \neq [\langle \vec{x}, \vec{\omega} \rangle]) \vee ([z] \neq [\langle \vec{y}, \vec{\omega} \rangle])$ goto 5a
   4.2b else if $([z'] \neq [\langle \vec{x}', \vec{\omega} \rangle]) \vee ([z'] \neq [\langle \vec{y}', \vec{\omega} \rangle])$
   goto 5b
   4.2c else goto 5c // $\mathbf{R}_1$ accepts h, h' with witness sk_1, sk_2
5a. // $\mathbf{R}_1$ does not accept h
   5a.1 $w_y' \leftarrow \mathsf{WExt}(td_{ext}, ([z], c_1, c_2), \pi; \xleftarrow{R})$
   5a.2 if $\neg\mathbf{R}_2(gpk, (([z], c_1, c_2)), w_y')$ return $\perp$
   5a.3 $\pi'' \leftarrow \mathsf{Prove}(gpk, crs, ([z''], c_1'', c_2''), w_y'; \xleftarrow{R})$
5b. repeat 5a with h' // $\mathbf{R}_1$ does not accept h'
5c. $\pi'' \leftarrow \mathsf{Prove}(gpk, crs, ([z''], c_1'', c_2''), (sk_1, sk_2); \xleftarrow{R})$
6. return $([z''], c_1'', c_2'', \pi'')$

Circuit $\mathrm{C}_{\mathrm{Map}}[gpk, crs, \vec{\omega}, sk_1](h_1, \ldots, h_\kappa)$:
1. for $i = 1 \ldots \kappa$
   1.1 if $\neg\mathsf{Val}_1(h_i)$ return $\perp$
   1.2 $([z_i], c_{i,1}, c_{i,2}, \pi_i) \leftarrow h_i$
   1.3 $\vec{x}_i \leftarrow \mathsf{Dec}(c_{i,1}, sk_1)$
2. $z \leftarrow \prod_{i=1}^{\kappa} \langle \vec{x}_i, \vec{\omega} \rangle \pmod p$
3. return $[z]$

Figure 3.2: Top: Circuit for addition of encodings. **Bottom**: Circuit implementing the multilinear map.

ADDITION. This section gives a description of Add_1 for adding level-1 encodings. The public parameters of the scheme contain an obfuscation of the circuit $\mathrm{C}_{\mathrm{Add}}$ shown in Figure 3.2 (top). Note that steps $5a$ or $5b$ are never reached with a binding crs' (but they may be reached with a hiding crs' later in the analysis). Add_1 runs the obfuscated circuit on the input encodings. The correctness of this algorithm follows from the correctness of Π, the completeness of Σ and the correctness, in our sense of (the possibly probabilistic) obfuscator Obf; see Section 3.2.2 for the definitions.

THE MULTILINEAR MAP. The multilinear map for Γ, on input κ encodings $h_i = ([z_i], c_{i,1}, c_{i,2}, \pi_i)$, uses sk_1 to recover the representation vectors $\vec{x}_i$. It then uses the explicit knowledge of $\vec{\omega}$ to compute the output of the map as

$$\mathsf{e}(h_1, \ldots, h_\kappa) := \left[\prod_{i=1}^{\kappa} \langle \vec{x}_i, \vec{\omega} \rangle \right].$$

The product in the exponent can be efficiently computed over $\mathbb{Z}_p$ for *any* polynomial level of linearity κ and any ℓ as it uses $\vec{x}_i$ and $\vec{\omega}$ explicitly. The κ-linearity of the map follows from the linearity of each of the multiplicands in the above product (and the completeness of Σ, the correctness of Π, and the correctness of the obfuscator Obf). An obfuscation $\overline{\mathrm{C}}_{\mathrm{Map}}$ of the circuit implementing this operation (see Figure 3.2,

bottom) will be made available through the public parameters and e is defined to run this circuit on its inputs.

SAMPLING. For sampling level-1 encodings, let $\vec{x}$ and $\vec{y}$ be vectors in $\mathbb{Z}_p^\ell$ satisfying $\langle \vec{x}, \vec{\omega} \rangle = \langle \vec{y}, \vec{\omega} \rangle$, set $[z] := [\langle \vec{y}, \vec{\omega} \rangle]$ (which can be computed using $[\vec{\omega}]$ and explicit knowledge of $\vec{x}$) and define the output of Sam_1 to be

$$h \leftarrow ([z], c_1 = \mathsf{Enc}(\vec{x}, pk_1; \vec{r}_1), c_2 = \mathsf{Enc}(\vec{y}, pk_2; \vec{r}_2), \\ \pi = \mathsf{Prove}(gpk, crs, ([z], c_1, c_2), (\vec{x}, \vec{y}, \vec{r}_1, \vec{r}_2); r).$$

More concretely, AFHLP set $\vec{x} = \vec{y} = (z, 0)$ when $\ell = 2$ and $\vec{x} = \vec{y} = (z, 0, 0)$ when $\ell = 3$. (These representations are called canonical.)

EXTRACTION. The extraction algorithm, on input $([z], c_1, c_2, \pi) \in S_1^{(z)}$, applies a universal hash function to $[z]$.

3.4 Our GES Construction

We now present our construction of a graded encoding scheme Γ according to the syntax introduced in Definition 2.3.4. We will use the following ingredients in our construction. A similar set of building blocks were used in [3].

1. A group setup algorithm $\mathsf{Setup}_{\mathbb{G}}(1^\lambda)$ that samples (the description of) a group $\mathbb{G}$, along with a random generator g of $\mathbb{G}$ and the group order p and the identity element 1.[3] We implicitly assume efficient algorithms for checking group membership, performing the group operation, inversion, and randomly sampling group elements. We further assume that every group element has a unique binary representation. We will also rely on a randomness extractor for this group.
2. A general-purpose probabilistic indistinguishability obfuscator **PIO** that we assume is secure against X-IND samplers.
3. A perfectly correct and IND-CPA-secure fully homomorphic PKE scheme Π with plaintext space $\mathbb{Z}_p^{\kappa+1}$.
4. An extractable dual-mode NIZK proof system Σ.
5. A language family Λ with hard membership problem and unique witnesses.

Given the above components, with formal syntax and security as defined in Definition 2.3.4, our graded encoding scheme Γ consists of the algorithms detailed in the sections that follow.

Following [42], we also use the bracket notation to compactly denote elements in $\mathbb{G}$. That is, we write $[z]$ and $[z']$ for two elements g^z and $g^{z'}$ in $\mathbb{G}$ and denote their product $g^z g^{z'}$ by $[z] + [z']$.

3.4.1 Setup

The Setup algorithm of Γ gets as input 1^λ and 1^κ. It samples $\mathcal{PP}_{\mathbb{G}} \leftarrow\!\!\$\ \mathsf{Setup}_{\mathbb{G}}(1^\lambda)$ with $\mathcal{PP}_{\mathbb{G}} := (\mathbb{G}, g, p, 1)$, generates two encryption key pairs $(pk_j, sk_j) \leftarrow\!\!\$\ \mathsf{Gen}(1^\lambda)$ for $j = 1, 2$, and an element $\omega \leftarrow\!\!\$\ \in \mathbb{Z}_p$. We will refer to $\mathbb{G}$ as the *base group*. It sets

$$[\vec{\omega}] := ([\omega], \ldots, [\omega^{s\kappa}]),$$

[3] It is conceivable that our security proofs also hold for non-prime p up to statistical defect terms related to randomization of elements modulo a composite number.

a vector of $s\kappa$ elements in the base group $\mathbb{G}$, with κ the number of desired levels and $s \in \mathbb{N}$ a system parameter. (Depending on s, the arising construction will have different properties. For $s \geq 1$, we get a construction in which the MDDH assumption holds; for $s \geq 2$, we get a construction in which additionally the RANK problem is hard.) It then samples $lk \leftarrow\!\!\$\ \mathsf{Gen}_{\mathsf{L}}(1^\lambda)$, and sets

$$gpk := (\mathcal{PP}_{\mathbb{G}}, pk_1, pk_2, [\vec{\omega}], lk).$$

We define $\mathbf{G}(1^\lambda)$ to be the randomized algorithm that runs the above steps and outputs gpk. This algorithm will be used to define the NIZK proof system.

The Setup algorithm then generates a *binding* CRS $(crs', td_{ext}) \leftarrow\!\!\$\ \mathsf{BCRS}(gpk)$, and also a *no-instance* of $\mathcal{L}_{lk}$ via $y \leftarrow\!\!\$\ \mathsf{NoSam}_{\mathsf{L}}(lk)$. It sets $crs := (crs', y)$. (The relation $\mathbf{R}$ that the NIZK should support will be defined shortly in Section 3.4.2.)

Finally, it constructs two obfuscated circuits $\overline{\mathrm{C}}_{\mathrm{Mult}}$ and $\overline{\mathrm{C}}_{\mathrm{Add}}$ of circuits $\mathrm{C}_{\mathrm{Mult}}$ and $\mathrm{C}_{\mathrm{Add}}$, which will be described in Section 3.4.3 and Section 3.4.4, respectively. Setup also selects a seed hk for a randomness extractor and outputs the scheme parameters

$$\mathcal{PP} := (gpk, crs, hk, \overline{\mathrm{C}}_{\mathrm{Add}}, \overline{\mathrm{C}}_{\mathrm{Mult}}).$$

3.4.2 Encodings and Equality

LEVEL-0 ENCODINGS. We treat algorithms for level-0 encodings separately in our construction as they behave somewhat differently to those from the other levels. For instance, when multiplied by other encodings, they do not result in an increase in encoding levels. The canonical choice for level-0 encodings is the ring $\mathbb{Z}_p$, which we adopt. These encodings, therefore, come with natural algorithms for generation, manipulation and testing of elements. Algorithm Mult when applied to inputs one of which is at level 0 corresponds to multiplication with the element in the zeroth level. The latter can in turn be implemented with a shift-and-add algorithm that employs the encoding addition Add of Section 3.4.3. We omit explicit mention of operations for level-0 encodings to ease notation and focus on the more interesting cases at levels 1 and above.[4]

LEVEL-κ ENCODINGS. We set $S_\kappa := \mathbb{G}$ in our scheme and use the algorithms associated with $\mathbb{G}$ for generation, equality testing, and addition of encodings at level κ. Once again, we omit these operations from the addition circuit for clarity. The multiplication circuit can only be called on a level-κ together with a level-0 encoding, which we have already excluded. However, we still have to deal with outputs at level κ in Mult.

OTHER LEVELS. For $0 < \ell < \kappa$ and $z \in \mathbb{Z}_p$, the encodings in $S_\ell^{(z)}$ consist of all tuples of the form

$$h := ([z], c_1, c_2, \pi, \ell),$$

where c_1, c_2 are two ciphertexts in the range of $\mathsf{Enc}(\cdot, pk_1)$ and $\mathsf{Enc}(\cdot, pk_2)$, respectively,[5] and π is a verifying NIZK proof under crs' that:

[4] We mention that previous GESs used more complex level-0 encodings, and since their encodings were *noisy*, they allowed only a limited number of operations on each encoding. Hence, implementing Mult on level-0 inputs via shift-and-add could be too costly in their settings.

[5] This "honest-ciphertext-generation" condition is necessary for the (bi)linearity of our addition and

(1) either c_1 and c_2 contain polynomials P_1 and P_2 of degree at most $s\ell$, such that $P_1(\omega) = P_2(\omega) = z$,
(2) or $y \in \mathcal{L}_{lk}$ (or both).

More formally, π must be a verifying proof that $(gpk, ([z], c_1, c_2, \ell))$ satisfies one relation $\mathbf{R}_1$ or $\mathbf{R}_2$ as follows.

Relation $\mathbf{R}_1$ on input gpk, an encoding $([z], c_1, c_2, \ell)$, and a witness $(P_1, P_2, \vec{r}_1, \vec{r}_2, sk_1, sk_2)$ accepts iff all of the following hold:

- $[z] \in \mathbb{G}$;
- both P_1 and P_2 are polynomials over $\mathbb{Z}_p$ of degree $\leq s\ell$ (given by their coefficient vectors);
- both P_1 and P_2 represent z in the sense that $[z] = [P_1(\omega)]$ and $[z] = [P_2(\omega)]$;
- both c_i are encryptions of (or decrypt to) P_i in the following sense:

$$\begin{aligned} &\text{for both } i \in \{1,2\} \ : \ c_i = \mathsf{Enc}(P_i, pk_i; \vec{r}_i) \\ &\qquad\qquad \vee \\ &\text{for both } i \in \{1,2\} \ : \ (pk_i, sk_i) = \mathsf{Gen}(sk_i) \wedge P_i = \mathsf{Dec}(c_i, sk_i). \end{aligned}$$

Note that there are two types of witnesses that can be used in proof generation for $\mathbf{R}_1$, namely $(P_1, P_2, \vec{r}_1, \vec{r}_2)$ and (sk_1, sk_2).

Let $\mathsf{R_L}$ be the relation for the trapdoor language Λ. Relation $\mathbf{R}_2$, given gpk, an encoding, and a witness w_y, accepts iff $\mathsf{R_L}(lk, y, w_y)$ accepts. (Note that the output of $\mathbf{R}_2$ is independent of input encodings.) Hence, intuitively, $\mathbf{R}_2$ provides an explicit trapdoor to simulate consistency proofs (in case $y \in \mathcal{L}_{lk}$).

We define $\mathbf{R} := \mathbf{R}_1 \vee \mathbf{R}_2$ and assume that Σ is a proof system with respect to $(\mathbf{G}, \mathbf{R})$ with $\mathbf{G}$ as defined in Section 3.4.1.

VALID AND CONSISTENT ENCODINGS. The following convention will be useful in the context of valid of encodings and the correctness of out scheme. We call an encoding h *valid* if the proof π verifies correctly under crs'. We write $\mathsf{Val}_\ell(h)$ iff h is valid and the level implicit in h matches ℓ. We call h *consistent* (with respect to gpk) if h is in the language defined by the first three conditions of relation $\mathbf{R}_1$ as well as the *first* clause of the disjunction above. (In particular, the corresponding ciphertexts $c[i]$ are possible outputs of $\mathsf{Enc}(P_i, pk_i)$; this implies that these ciphertexts behave as expected under the homomorphic evaluation algorithm Eval.) Note that consistency implies validity but the converse is not necessarily the case and hence a valid encoding may not lie in any S_ℓ. For example this would be the case if an "anomalous" ciphertext *decrypts* correctly to a valid representation, but does not lie in the range of Enc. Furthermore, validity can be publicly and efficiently checked, while this is *not* necessarily the case for consistency. We note, however, that if the encryption scheme does not allow for anomalous ciphertexts, our GES would also have efficiently recognizable encodings. We leave the construction of such FHE schemes as an open problem.

ALGORITHM Eq. The equality algorithm Eq_ℓ returns 1 iff the first components of the inputs match. The correctness of this algorithm follows from the fact that the base group $\mathbb{G}$ has unique representations. (Recall from GES syntax that Eq_ℓ is only required to work with respect to consistent encodings.)

multiplication algorithms. Unfortunately, this also prevents the sets $S_\ell^{(z)}$ from being efficiently recognizable.

Circuit $\mathrm{C}_{\mathrm{Add}}[gpk, crs, sk_1, sk_2, td_{ext}](\ell, h, h')$: // for $1 \le \ell \le \kappa - 1$

1. if $\neg(\mathsf{Val}_\ell(h) \wedge \mathsf{Val}_\ell(h'))$ then return $\perp$
2. parse $([z], c_1, c_2, \pi, \ell) \leftarrow h$ and $([z'], c_1', c_2', \pi', \ell) \leftarrow h'$
3. $[z''] \leftarrow [z] + [z']$; $c_1'' \leftarrow c_1 + c_1'$; $c_2'' \leftarrow c_2 + c_2'$
4. $P_1 \leftarrow \mathsf{Dec}(c_1, sk_1)$; $P_2 \leftarrow \mathsf{Dec}(c_2, sk_2)$
   $P_1' \leftarrow \mathsf{Dec}(c_1', sk_1)$; $P_2' \leftarrow \mathsf{Dec}(c_2', sk_2)$
5. if $[z] \neq [P_1(\omega)] \vee [z] \neq [P_2(\omega)] \vee [z'] \neq [P_1'(\omega)] \vee [z'] \neq [P_2'(\omega)]$ then
   5.1. $w_y' \leftarrow\!\!\$\ \mathsf{WExt}(td_{ext}, ([z], c_1, c_2), \pi)$
   5.2. if $\neg \mathbf{R}_2(gpk, ([z], c_1, c_2, \ell), w_y')$ then return $\perp$
   5.3. $\pi'' \leftarrow\!\!\$\ \mathsf{Prove}(gpk, crs, ([z''], c_1'', c_2''), w_y')$
6. else $\pi'' \leftarrow\!\!\$\ \mathsf{Prove}(gpk, crs, ([z''], c_1'', c_2''), (sk_1, sk_2))$
7. return $([z''], c_1'', c_2'', \pi'', \ell)$

Figure 3.3: The probabilistic circuit used to add encodings for levels $1 \le \ell \le \kappa - 1$. The checks at 5 are never passed in an honest execution of the protocol. We emphasize that the test in step 5 is implemented using the values $[\omega^i]$. The random coins needed for randomized operations are internally generated after obfuscating with **PIO**.

POLYNOMIAL REPRESENTATIONS. A significant conceptual difference with the work of AFHLP is that we represent exponents in $\mathbb{Z}_p$ with polynomials instead of vectors. This generalization enables natural notion of levels corresponding to the degrees of the representing polynomials. We observe that, a level-ℓ encoding h is not a *valid* level-ℓ' encoding if $\ell' \neq \ell$ as the perfectly sound proof π included in h depends on the instance and in particular on the level.

3.4.3 Addition

We now provide a procedure for adding two level-ℓ encodings $h = ([z], c_1, c_2, \pi, \ell)$ and $h' = ([z'], c_1', c_2', \pi', \ell)$ in S_ℓ. Conceptually, our addition circuit operates similarly to that of AFHLP. The main difference is that encodings contain polynomials and the levels. We exploit the structure of the base group as well as the homomorphic properties of the encryption scheme to "add together" the first and second components of the inputs. We then use (sk_1, sk_2) as a witness to generate a proof π'' that the new tuple is well formed. For technical reasons we check both the validity of h and h' (by checking π and π') and their consistency (using (sk_1, sk_2)).

Figure 3.3 details the operation of the addition circuit $\mathrm{C}_{\mathrm{Add}}$. A **PIO** of this circuit will be made public via the parameters $\mathcal{PP}$. We emphasize that step 5, that is, the explicit consistency check, is never reached under a binding crs' (due to the perfect soundness of the proof system), but they may be reached with a hiding crs' later in the security analysis. Let us expand on this.

In the analysis, we need to specify how $\mathrm{C}_{\mathrm{Add}}$ behaves if it encounters valid inputs (in the sense the proofs pass NIZK verification), but nevertheless are inconsistent in the sense that at least one of encodings does not decrypt to a valid representation. Let us call such inputs *bad*.

With the knowledge of secret keys, such bad inputs can be recognized, and the natural choice would be to define $\mathrm{C}_{\mathrm{Add}}$ to abort when this is the case. With this choice, however, we run into the following problem. During the security proof we will set the addition circuit to answer all valid inputs (including bad ones) with

simulated proofs. On the other hand, the original addition circuit rejects such inputs. (Furthermore, it cannot even simulate proofs for wrong statements, and hence cannot answer bad inputs with valid-looking proofs.)

On a high level, we would like to modify how $\mathrm{C}_{\mathrm{Add}}$ reacts on bad inputs so that it uses a NIZK simulation trapdoor on bad inputs. The difficulty with this strategy is that no such simulation trapdoor exists when the NIZK CRS is binding. Hence, we create our own NIZK trapdoor through an extra "OR branch" in the proved statement (akin to the Feige–Lapidot–Shamir transform). This gives us a little more flexibility in defining and using that trapdoor.

More specifically, recall that our CRS is of the form $crs = (crs', y)$ where crs' is a binding CRS for the dual-mode NIZK proof system, and y is a no-instance of $\mathcal{L}_{lk}$. However our actual means to fake proofs will be to switch y to a yes-instance and use a witness w_y to produce proofs. Specifically, in the security proof, we will eventually let $\mathrm{C}_{\mathrm{Add}}$ use a simulation trapdoor w_y (instead of a simulation trapdoor for the NIZK). The benefit of this is that $\mathrm{C}_{\mathrm{Add}}$ will know an extraction trapdoor td'_{ext} (that of course only exists if the CRS crs' is in the binding mode) which it can use to extract a witness from a given proof π. Thus, whenever $\mathrm{C}_{\mathrm{Add}}$ encounters a bad input, it can extract a witness w'_y, which *must* at that point be a simulation trapdoor w_y. This simulation trapdoor w_y can then immediately be used to produce a fake proof π'' even upon bad inputs. In other words, $\mathrm{C}_{\mathrm{Add}}$ knows no simulation trapdoor a priori, but it can extract one from any simulated proof for a false statement.

The Add_ℓ algorithm simply runs the obfuscated circuit on the input encodings and ℓ. The correctness of this algorithm follows from that of Π, the completeness of Σ and the correctness, in our sense, of the (probabilistic) obfuscator **PIO**. Note that FHE correctness is only guaranteed to hold with respect to ciphertexts that are in the range of encryption or evaluation (and not necessarily for anomalous ones that decrypt correctly). This, in particular, means that we cannot enlarge the set of encodings to contain all valid ones (as opposed to just consistent ones) to get efficient decidability of encoding sets as correctness can no longer be established. (See also remark on validity on page 37.) Note that full compactness ensures that the ciphertexts output by Add_ℓ are in the range of encryption, and hence they can be further operated on with Eval.

3.4.4 Multiplication

Given two encodings $h = ([z], c_1, c_2, \pi, \ell)$ and $h' = ([z'], c'_1, c'_2, \pi', \ell')$ at levels ℓ and ℓ' respectively, the multiplication algorithms operates analogously to addition as follows. The corresponding circuit $\mathrm{C}_{\mathrm{Mult}}$ has both decryption keys and now also $\omega \in \mathbb{Z}_p$ hardwired in. After validity checks and decrypting the input ciphertexts, it performs the multiplication of the polynomials encrypted under c_i and c'_i homomorphically using a convolution operation on the coefficient vectors. However, it cannot obviously compute the element $[zz']$ in the base group $\mathbb{G}$. Suppose c_1 and c'_1 encrypt polynomials P and P' of degrees at most $s\ell$ and $s\ell'$ respectively and such that $[z] = [P(\omega)]$ and $[z'] = [P'(\omega)]$. The multiplication circuit uses the explicit knowledge of ω and polynomials P and P' to compute $[zz'] = [(P * P')(\omega)]$.[6] Circuit

[6] Observe that with the explicit knowledge of $P * P'$ and the powers $([\omega^i])_{1 \le i \le s\kappa}$ it is also possible to compute $[zz']$ as long as $P * P'$ is of degree $\le s\kappa$; this will be exploited in the security analysis in

Circuit $\mathrm{C}_{\mathrm{Mult}}[gpk, crs, \omega, sk_1, sk_2, td_{ext}](\ell, \ell', h, h')$: // for $1 \le \ell, \ell' \le \kappa - 1$
1. if $\neg(\mathsf{Val}_\ell(h) \wedge \mathsf{Val}_{\ell'}(h')) \vee \ell + \ell' > \kappa$ then return $\perp$
2. parse $([z], c_1, c_2, \pi, \ell) \leftarrow h$ and $([z'], c_1', c_2', \pi', \ell') \leftarrow h'$
3. $c_1'' \leftarrow c_1 * c_1'$; $c_2'' \leftarrow c_2 * c_2'$
4. $P_1 \leftarrow \mathsf{Dec}(c_1, sk_1)$; $P_2 \leftarrow \mathsf{Dec}(c_2, sk_2)$
   $P_1' \leftarrow \mathsf{Dec}(c_1', sk_1)$; $P_2' \leftarrow \mathsf{Dec}(c_2', sk_2)$
5. $z'' \leftarrow (P_1 * P_1')(\omega)$
6. if $[z] \neq [P_1(\omega)] \vee [z] \neq [P_2(\omega)] \vee [z'] \neq [P_1'(\omega)] \vee [z'] \neq [P_2'(\omega)]$ then
   6.1. $w_y' \leftarrow\!\!\$\ \mathsf{WExt}(td_{ext}, ([z], c_1, c_2), \pi)$
   6.2. if $\neg \mathbf{R}_2(gpk, ([z], c_1, c_2), w_y')$ then return $\perp$
   6.3. $\pi'' \leftarrow\!\!\$\ \mathsf{Prove}(gpk, crs, ([z''], c_1'', c_2''), w_y')$
7. else $\pi'' \leftarrow\!\!\$\ \mathsf{Prove}(gpk, crs, ([z''], c_1'', c_2''), (sk_1, sk_2))$
8. If $(\ell + \ell' = \kappa)$ then return $[z'']$ else return $([z''], c_1'', c_2'', \pi'', \ell + \ell')$

Figure 3.4: Circuit used for multiplying encodings for levels $1 \le \ell, \ell' \le \kappa - 1$. Step 6 is never reached in an honest execution of the protocol with a binding crs. The random coins needed for randomized operations are internally generated after obfuscating with **PIO**.

$\mathrm{C}_{\mathrm{Mult}}$ is shown in Figure 3.4. Note that similarly to addition, step 6 performs explicit checks of consistency of encodings that will only be used in the analysis under a hiding crs'.

The correctness of these maps follows from the correctness of Π and **PIO**, and the completeness of Σ.

ENABLING GRADED MULTIPLICATION. The main difference between our circuit $\mathrm{C}_{\mathrm{Mult}}$ and that of [3] is that here we need to output auxiliary information (c_1, c_2, π) for multiplied encodings at output levels below κ. This information allows the multiplication algorithm to operate in a graded fashion as any output encoding by $\mathrm{C}_{\mathrm{Mult}}$ can be fed back into $\mathrm{C}_{\mathrm{Mult}}$ as long as it lies at a level $\ell < \kappa$.[7] In order to enable $\mathrm{C}_{\mathrm{Mult}}$ to generate this auxiliary information, we use an encryption scheme that is also homomorphic with respect to multiplication in the plaintext ring. In contrast, AFHLP only rely on an additively homomorphic encryption scheme.

3.4.5 Sampling

Given polynomials P_1 and P_2 of degree at most $s\ell$ and satisfying $P_1(\omega) = P_2(\omega) = z$ we can generate an encoding from $S_\ell^{(z)}$ by computing

$$h \leftarrow ([z], c_1 = \mathsf{Enc}(P_1, pk_1; \vec{r}_1), c_2 = \mathsf{Enc}(P_2, pk_2; \vec{r}_2), \\ \pi = \mathsf{Prove}(gpk, crs, ([z]_i, c_1, c_2, \ell), (P_1, P_2, \vec{r}_1, \vec{r}_2); r), \ell). \tag{3.2}$$

Hence, our sampling algorithm $\mathsf{Sam}_\ell(z)$ sets $P_1(X) = P_2(X) = z \in \mathbb{Z}_p$ and computes an encoding through Equation (3.2). We call these the *canonical* encodings of z, independently of ℓ. We note that this procedure is that in [3] adapted to the generalized notion of polynomial representations.

Section 3.6.

[7] Recall that encodings at level κ can only be multiplied with level-0 encodings, i.e., with elements in $\mathbb{Z}_p$.

$\underline{\kappa\text{-Switch}}^{\mathcal{A}}_{\Gamma}(\lambda)$:
$(\mathcal{PP};\omega) \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda, 1^\kappa)$ // ω generated within Setup
$((P_{0,1}, P_{0,2}), (P_{1,1}, P_{1,2}), \ell, st) \leftarrow\!\!\$\ \mathcal{A}_1(\mathcal{PP}, \omega)$
$b \leftarrow\!\!\$\ \{0,1\}$; $\vec{r}_1, \vec{r}_2 \leftarrow\!\!\$\ \{0,1\}^{\mathrm{rl}(\lambda)}$
$c_1 \leftarrow \mathsf{Enc}(P_{b,1}, pk_1; \vec{r}_1)$; $c_2 \leftarrow \mathsf{Enc}(P_{b,2}, pk_2; \vec{r}_2)$
$\pi \leftarrow\!\!\$\ \mathsf{Prove}(gpk, crs, ([P_{b,1}(\omega)], c_1, c_2, \ell), (P_{b,1}, P_{b,2}, \vec{r}_1, \vec{r}_2))$
$h_b \leftarrow ([P_{b,1}(\omega)], c_1, c_2, \pi, \ell)$
$b' \leftarrow\!\!\$\ \mathcal{A}_2(h_b, st)$
Return $(b = b')$

Figure 3.5: Game formalizing the indistinguishability of encodings. (This game is specific to our construction Γ from Section 3.4.) An adversary is legitimate if it outputs polynomials such that $P_{0,1}(\omega) = P_{0,2}(\omega) = P_{1,1}(\omega) = P_{1,2}(\omega)$ of degree at most $s\ell$. We note that $\mathcal{A}$ gets explicit access to secret exponent ω generated at setup. Here $\mathrm{rl}(\lambda)$ is a polynomial indicating the length of the random coins used by the encryption algorithm.

3.4.6 Extraction

Since at each level ℓ the first component $[z]$ is unique for each set $S_\ell^{(z)}$, we may extract a uniform string from $h = ([z], c_1, c_2, \pi, \ell)$ for a uniform z by applying a randomness extractor seeded with hk to $[z]$.

3.5 Indistinguishability of Encodings

We show that a key property used by AFHLP in the analysis of their multilinear map [3, Theorem 5.3] is also exhibited by our graded scheme. Roughly speaking, this property states that for any given level ℓ, any two valid encodings of the same $\mathbb{Z}_p$-element are computationally indistinguishable. This claim is formalized via the κ-Switch game shown in Figure 3.5. Note that in this game, we allow the adversary to not only choose the representation polynomials, but also let him see part of the private information not available through the public parameters, namely the exponent ω.

Theorem 3.5.1 (Encoding switch)**.** *Let Γ be the GES constructed in Section 3.4 with respect to an X-IND-secure probabilistic obfuscator* **PIO***, an* IND-CPA*-secure encryption scheme Π, a dual-mode NIZK proof system Σ, and a language family Λ. Then, encodings of the same ring element $z \in \mathbb{Z}_p$ are indistinguishable at all levels. More precisely, for any legitimate* PPT *adversary $\mathcal{A}$ there are* PPT *adversaries $\mathcal{B}_1$, $\mathcal{B}_2$, $\mathcal{B}_3$ and $\mathcal{B}_4$ of essentially the same complexity as $\mathcal{A}$ such that for all $\lambda \in \mathbb{N}$*

$$\mathbf{Adv}^{\kappa\text{-switch}}_{\Gamma,\mathcal{A}}(\lambda) \leq 3 \cdot \left(\mathbf{Adv}^{\mathrm{mem}}_{\Lambda,\mathcal{B}_1}(\lambda) + 6 \cdot \mathbf{Adv}^{}_{\mathbf{PIO},\mathcal{B}_2}(\lambda) + \mathbf{Adv}^{\mathrm{crs}}_{\Sigma,\mathcal{B}_3}(\lambda)\right) + 2 \cdot \mathbf{Adv}^{\text{ind-cpa}}_{\Pi,\mathcal{B}_4}(\lambda).$$

The proof of this result follows largely that in [3] and we include it for completeness. The main difference is that we have to deal with obfuscations of the new multiplication circuit.

Outline. We proceed via a sequence of 5 games, starting with κ-Switch and ending in a game where the challenge encoding is independent of the bit b. Figure 3.6

shows the steps used in the proof of the theorem. We use helper Lemma 3.5.2 for changing the addition and multiplication circuits to "forget" (one or both) the secret keys and the extraction trapdoor. We now justify each of these steps in more detail below. We let W_i denote the event that Game_i outputs 1.

Game	crs'	y	$\mathrm{C}_{\mathrm{Add}}$ knows	$\mathrm{C}_{\mathrm{Mult}}$ knows	c_1 contains	c_2 contains	**Remark**
0	binding	$\notin \mathcal{L}_{lk}$	sk_1, sk_2, td_{ext}	$\underline{sk_1}, sk_2, td_{ext}$	$P_{b,1}$	$P_{b,2}$	
1	hiding	$\in \mathcal{L}_{lk}$	w_y	$\underline{sk_1}, w_y$	$P_{b,1}$	$P_{b,2}$	Lemma 3.5.2 ($i = 1$)
2	hiding	$\in \mathcal{L}_{lk}$	w_y	$\underline{sk_1}, w_y$	$P_{b,1}$	$P_{1,2}$	IND-CPA wrt. pk_2
3	binding	$\notin \mathcal{L}_{lk}$	sk_1, sk_2, td_{ext}	$\underline{sk_1}, sk_2, td_{ext}$	$P_{b,1}$	$P_{1,2}$	Lemma 3.5.2 (reverse, $i = 1$)
4	hiding	$\in \mathcal{L}_{lk}$	w_y	$\underline{sk_2}, w_y$	$P_{b,1}$	$P_{1,2}$	Lemma 3.5.2 ($i = 2$)
5	hiding	$\in \mathcal{L}_{lk}$	w_y	$\underline{sk_2}, w_y$	$P_{1,1}$	$P_{1,2}$	IND-CPA wrt. pk_1
							Encoding indep. of b

Figure 3.6: Outline of the proof steps of Theorem 3.5.1. The underlined secret key in the "$\mathrm{C}_{\mathrm{Mult}}$ knows" column indicates the key that is used in decryption to construct $[z'']$. For instance, in Game_0, key sk_1 is used to obtain P_1 and P_1', which are then used to compute $[z''] = [(P_1 * P_1')(\omega)]$ within $\mathrm{C}_{\mathrm{Mult}}$.

Game_0: This is the κ-Switch game (see Figure 3.5) with a binding crs' and $y \notin \mathcal{L}_{lk}$. The addition and multiplication circuits are defined in Figure 3.3 and Figure 3.4, respectively.

Game_1: We change the public parameters so that they include a *hiding* crs', a yes instance y via $\mathsf{YesSam}_{\mathsf{L}}(lk)$ and obfuscations of circuits $\widehat{\mathrm{C}}_{\mathrm{Add}}$ and $\widehat{\mathrm{C}}^{(1)}_{\mathrm{Mult}}$ (see Figure 3.7). Thus, the second circuit uses sk_1 to decrypt the first ciphertexts given as inputs. Observe that these circuits use the witness w_y to $y \in \mathcal{L}_{lk}$ to produce the output proofs π'', and therefore the *simultaneous* knowledge of decryption keys sk_1, sk_2 is no longer needed. By Lemma 3.5.2 we have that

$$|\Pr[\mathrm{W}_0(\lambda)] - \Pr[\mathrm{W}_1(\lambda)]| \leq \mathbf{Adv}^{\mathrm{mem}}_{\Lambda,\mathcal{B}_1}(\lambda) + 6 \cdot \mathbf{Adv}^{}_{\mathbf{PIO},\mathcal{B}_2}(\lambda) + \mathbf{Adv}^{\mathrm{crs}}_{\Sigma,\mathcal{B}_3}(\lambda).$$

Game_2: This game generates the second challenge ciphertext c_2 by encrypting polynomial $P_{1,2}$ *even when* $b = 0$. We bound this transition via the IND-CPA security of Π with respect to pk_2. The reduction will choose a first decryption key sk_1 and a witness w_y so as to be able to construct $\widehat{\mathrm{C}}^{(1)}_{\mathrm{Mult}}$. It will also generate a NIZK simulation trapdoor td_{zk} (recall the CRS is in the hiding mode) to construct simulated proofs π for the (inconsistent) challenge encoding h_b. Note that the perfect ZK property guarantees that these proofs are identically distributed to the real ones in Game_1. More detailed, in Game_1 the polynomial $P_{1,2}$ is encrypted under pk_2 regardless of the value of the bit b. Thus, on $\mathcal{A}_1$'s response $((P_{0,1}, P_{0,2}), (P_{1,1}, P_{1,2}), \ell, st)$, the game sets $c_1 \leftarrow \mathsf{Enc}(P_{b,1}, pk_1)$ for a random bit b, and $c_2 \leftarrow \mathsf{Enc}(P_{1,2}, pk_2)$. We claim that

$$|\Pr[\mathrm{W}_1(\lambda)] - \Pr[\mathrm{W}_2(\lambda)]| \leq \mathbf{Adv}^{\text{ind-cpa}}_{\Pi,\mathcal{B}_4}(\lambda).$$

Consider a PPT distinguisher $\mathcal{B}_4$ against the IND-CPA security of scheme Π (with respect to key pair (sk_2, pk_2)) as follows. The distinguisher runs Game_1 and uses the adversary $\mathcal{A}$ as a subroutine. When it receives $\mathcal{A}_1$'s outputs $((P_{0,1}, P_{0,2}), (P_{1,1}, P_{1,2}), \ell, st)$, $\mathcal{B}_4$ generates $c_1 \leftarrow\!\!\$\ \mathsf{Enc}(P_{b,1}, pk_1)$ for a random bit b. It then submits $(P_{b,2}, P_{1,2})$ to its IND-CPA challenger and gets back a challenge c^*. It sets $c_2 := c^*$. The proof π on the instance $x := ([z], c_1, c_2, \ell)$ is generated using the simulation trapdoor of the proof system guaranteed by the zero-knowledge property. (Note that in contrast to the Naor–Yung paradigm we do *not* prove an invalid statement and do not need to rely on simulation soundness.) Namely, $\pi \leftarrow\!\!\$\ \mathsf{Sim}(td_{zk}, x)$. Finally, $\mathcal{B}_4$ sets $h := ([z], c_1, c_2, \pi, \ell)$ and runs $\mathcal{A}_2(h, st)$ to get a bit b'. It returns $(b = b')$. Game_1 and Game_2 differ only in how c_2 and π for the challenge encoding are generated. First note that real and simulated proofs are *identically* distributed under the hiding crs'. Second, letting d denote the IND-CPA challenge bit, when $d = 0$ ciphertext c_2 encrypts $P_{b,2}$ and $\mathcal{B}_4$ perfectly simulates Game_1 for $\mathcal{A}$, and when $d = 1$ ciphertext c_2 encrypts $P_{1,2}$ and $\mathcal{B}_4$ perfectly simulates Game_2.

Game_3: The public parameters are changed back so that they include a binding crs', a no-instance $y \leftarrow\!\!\$\ \mathsf{NoSam}_{\mathsf{L}}(lk)$ and obfuscations of circuits $\mathrm{C}_{\mathrm{Add}}$ and $\mathrm{C}_{\mathrm{Mult}}$ of Figure 3.3 and Figure 3.4. Once again by Lemma 3.5.2 (in the reverse direction and with $i = 1$) we have that

$$|\Pr[\mathrm{W}_2(\lambda)] - \Pr[\mathrm{W}_3(\lambda)]| \leq \mathbf{Adv}^{\mathrm{mem}}_{\Lambda,\mathcal{B}_1}(\lambda) + 6 \cdot \mathbf{Adv}^{}_{\mathbf{PIO},\mathcal{B}_2}(\lambda) + \mathbf{Adv}^{\mathrm{crs}}_{\Sigma,\mathcal{B}_3}(\lambda).$$

Circuit $\widehat{\mathrm{C}}_{\mathrm{Add}}[gpk, crs, w_y](\ell, h, h')$:
1. if $\neg(\mathsf{Val}_\ell(h) \wedge \mathsf{Val}_\ell(h'))$ then return $\perp$
2. parse $([z], c_1, c_2, \pi, \ell) \leftarrow h$, and $([z'], c'_1, c'_2, \pi', \ell) \leftarrow h'$
3. $[z''] \leftarrow [z] + [z']$; $c''_1 \leftarrow c_1 + c'_1$; $c''_2 \leftarrow c_2 + c'_2$
4. // omitted: depends on sk_1 and sk_2
5. $\pi'' \leftarrow\!\!\$\ \mathsf{Prove}(gpk, crs, ([z''], c''_1, c''_2, \ell), w_y)$
6. // omitted: depends on sk_1 and sk_2
7. return $([z''], c''_1, c''_2, \pi'', \ell)$

Circuit $\widehat{\mathrm{C}}^{(i)}_{\mathrm{Mult}}[gpk, crs, \omega, sk_i, w_y](\ell, \ell', h, h')$:
1. if $\neg(\mathsf{Val}_\ell(h) \wedge \mathsf{Val}_{\ell'}(h')) \vee \ell + \ell' > \kappa$ then return $\perp$
2. parse $([z], c_1, c_2, \pi, \ell) \leftarrow h$ and $([z'], c'_1, c'_2, \pi', \ell') \leftarrow h'$
3. $c''_1 \leftarrow c_1 \cdot c'_1$; $c''_2 \leftarrow c_2 \cdot c'_2$
4. $P_i \leftarrow \mathsf{Dec}(c_i, sk_i)$; $P'_i \leftarrow \mathsf{Dec}(c'_i, sk_i)$ // depends on sk_i only
5. $z'' \leftarrow (P_i * P'_i)(\omega)$
6. $\pi'' \leftarrow\!\!\$\ \mathsf{Prove}(gpk, crs, ([z''], c''_1, c''_2, \ell + \ell'), w_y)$
7. // omitted: depends on sk_1 and sk_2
8. If $(\ell + \ell' = \kappa)$ then return $[z'']$ else return $([z''], c''_1, c''_2, \pi'', \ell + \ell')$

Figure 3.7: Top: Circuit $\widehat{\mathrm{C}}_{\mathrm{Add}}$ where witness w_y to $y \in \mathcal{L}_{lk}$ is used to produce π''. Note that the secret keys (sk_1, sk_2) or the extraction trapdoor td_{ext} are no longer used by this circuit. **Bottom:** Circuits $\widehat{\mathrm{C}}^{(i)}_{\mathrm{Mult}}$ were only one key sk_i is used to decrypt P_i and P'_i and witness w_y to $y \in \mathcal{L}_{lk}$ is used to produce π''. The secret key $sk[3-i]$ and the extraction trapdoor td_{ext} are not used by this circuit.

Game_4**:** The public parameters are changed so that they include a hiding crs', a yes-instance $y \leftarrow\!\!\$\ \mathsf{YesSam}_{\mathsf{L}}(lk)$ and obfuscations of circuits $\widehat{\mathrm{C}}^{(2)}_{\mathrm{Mult}}$ and $\widehat{\mathrm{C}}_{\mathrm{Add}}$ (see Figure 3.7). Observe that knowledge of sk_1 is no longer needed. Similarly to Game_1, by Lemma 3.5.2 with $i = 2$ we have that

$$|\Pr[\mathrm{W}_3(\lambda)] - \Pr[\mathrm{W}_4(\lambda)]| \leq \mathbf{Adv}^{\mathrm{mem}}_{\Lambda,\mathcal{B}_1}(\lambda) + 6 \cdot \mathbf{Adv}_{\mathbf{PIO},\mathcal{B}_2}(\lambda) + \mathbf{Adv}^{\mathrm{crs}}_{\Sigma,\mathcal{B}_3}(\lambda).$$

Game_5**:** This transitions is defined analogously to that introduced in Game_2. The polynomial encrypted under public key pk_1 is $P_{1,1}$ regardless of the bit b. Thus, after receiving $((P_{0,1}, P_{0,2}), (P_{1,1}, P_{1,2}), \ell, st)$ from $\mathcal{A}_1$, the game sets $c_1 \leftarrow \mathsf{Enc}(P_{1,1}, pk_1)$, and $c_2 \leftarrow \mathsf{Enc}(P_{1,2}, pk_2)$. Using a similar argument to that for Game_2 we get that

$$|\Pr[\mathrm{W}_4(\lambda)] - \Pr[\mathrm{W}_5(\lambda)]| \leq \mathbf{Adv}^{\text{ind-cpa}}_{\Pi,\mathcal{B}_4}(\lambda).$$

Finally, note that the challenge encoding in Game_5 is independent of the random bit b and the advantage of any (even unbounded) adversary $\mathcal{A}$ is identically 0. □

We now prove the helper lemma that we relied upon in the above.

Lemma 3.5.2 (Forgetting secret keys)**.** *Let Γ be the GES from Section 3.4 with respect to an X-*IND*-secure probabilistic obfuscator* **PIO***, an* IND-CPA*-secure encryption scheme Π, a dual-mode NIZK proof system Σ, and a language family Λ. For $i = 1, 2$, consider the modified parameter generation algorithm $\mathsf{Setup}^{(i)}$ that samples*

a yes-instance $y \in \mathcal{L}_{lk}$ *and outputs obfuscations of the circuits* $\widehat{\mathrm{C}}_{\mathrm{Add}}$ *and* $\widehat{\mathrm{C}}^{(i)}_{\mathrm{Mult}}$ *shown in Figure 3.7. Let*

$$\mathbf{Adv}^{\kappa\text{-forget}}_{\Gamma,i,\mathcal{A}}(\lambda) : GES := 2 \cdot \Pr\left[\mathcal{PP}_0 \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda, 1^\kappa);\ \mathcal{PP}_1 \leftarrow\!\!\$\ \mathsf{Setup}^{(i)}(1^\lambda, 1^\kappa);\right.$$
$$\left. b \leftarrow\!\!\$\ \{0,1\}; b' \leftarrow\!\!\$\ \mathcal{A}(\mathcal{PP}_b)\ :\ b = b'\right] - 1.$$

Then, for any $i \in \{1, 2\}$ *and any* PPT *adversary* $\mathcal{A}$ *there are* PPT *adversaries* $\mathcal{B}_1, \mathcal{B}_2$ *and* $\mathcal{B}_3$ *of essentially the same complexity as* $\mathcal{A}$ *such that for all* $\lambda \in \mathbb{N}$

$$\mathbf{Adv}^{\kappa\text{-forget}}_{\Gamma,i,\mathcal{A}}(\lambda) \leq \mathbf{Adv}^{\mathrm{mem}}_{\Lambda,\mathcal{B}_1}(\lambda) + 6 \cdot \mathbf{Adv}_{\mathbf{PIO},\mathcal{B}_2}(\lambda) + \mathbf{Adv}^{\mathrm{crs}}_{\Sigma,\mathcal{B}_3}(\lambda).$$

Proof. We provide an outline of the game hops in Figure 3.8 and give the details next.

Game_0**:** We start with a game that runs $\mathcal{A}$ on $\mathcal{PP}_0$; that is with an obfuscation of $\mathrm{C}_{\mathrm{Add}}$ and $\mathrm{C}_{\mathrm{Mult}}$ (see Figure 3.3 and Figure 3.4), and a no-instance $y \notin \mathcal{L}_{lk}$.

Game_1**:** Our first change consists in modifying the obfuscated $\mathrm{C}_{\mathrm{Mult}}$ so that in step 5 it uses P_i and P'_i (instead of P_1 and P'_1) to construct $[z'']$. (Both keys are still needed in step 4.) Note there is *no change* when $i = 1$, but when $i = 2$ we show this modification leads to a functionally equivalent circuit. Indeed, since the NIZK proof system is perfectly sound (the crs' is binding) and $y \notin \mathcal{L}_{lk}$, any valid encoding must satisfy $P_1(\omega) = P_i(\omega)$. Hence, using (P_i, P'_i) instead of (P_1, P'_1) leads to the same circuit outputs. The security of the obfuscator can be used to bound the difference in the outputs of Game_0 and Game_1.

Game_2**:** We sample $y \in \mathcal{L}_{lk}$ instead of $y \notin \mathcal{L}_{lk}$. By the hardness of deciding membership for $\mathcal{L}_{lk}$, this only negligibly changes the game's output.

Game_3**:** We hardwire the witness w_y to $y \in \mathcal{L}_{lk}$ in $\mathrm{C}_{\mathrm{Add}}$ and $\mathrm{C}_{\mathrm{Mult}}$, and remove td_{ext} from both circuits. We claim that this change does not change the functionality of $\mathrm{C}_{\mathrm{Add}}$ and $\mathrm{C}_{\mathrm{Mult}}$ at all. To see this, recall that $\mathcal{L}_{lk}$ has unique witnesses. Hence, any witness w'_y extracted by $\mathrm{C}_{\mathrm{Add}}$ or $\mathrm{C}_{\mathrm{Mult}}$ in Game_2 must be equal to the hardwired witness w_y in Game_3. Since crs' is binding, extraction will always succeed in Game_2 (if it comes to step 5.1 in $\mathrm{C}_{\mathrm{Add}}$ or step 6.1 in $\mathrm{C}_{\mathrm{Mult}}$). Thus this transition can be justified by the security of the obfuscator (for two circuits).

Game_4**:** The string crs' included in the public parameters is changed to the hiding mode. Hence proofs generated under crs' will be perfectly witness indistinguishable in this game. This hop can be justified by the CRS indistinguishability of the dual-mode NIZK proof system.

Game	crs'	y	$\mathrm{C}_{\mathrm{Add}}$ knows	$\mathrm{C}_{\mathrm{Mult}}$ knows	π''-witness	**Remark**
0	binding	$\notin \mathcal{L}_{lk}$	sk_1, sk_2, td_{ext}	$\underline{sk_1}, sk_2, td_{ext}$	(sk_1, sk_2) or w'_y	
1	binding	$\notin \mathcal{L}_{lk}$	sk_1, sk_1, td_{ext}	$\underline{sk_i}, sk[3-i], td_{ext}$	(sk_1, sk_2) or w'_y	**PIO**/soundness
2	binding	$\in \mathcal{L}_{lk}$	sk_1, sk_1, td_{ext}	$\underline{sk_i}, sk[3-i], td_{ext}$	(sk_1, sk_2) or w'_y	$\mathcal{L}_{lk}$ hard
3	binding	$\in \mathcal{L}_{lk}$	sk_1, sk_2, w_y	$\underline{sk_i}, sk[3-i], w_y$	(sk_1, sk_2) or w_y	**PIO**/unique w_y
4	hiding	$\in \mathcal{L}_{lk}$	sk_1, sk_2, w_y	$\underline{sk_i}, sk[3-i], w_y$	(sk_1, sk_2) or w_y	CRS indist.
5	hiding	$\in \mathcal{L}_{lk}$	w_y	$\underline{sk_i}, w_y$	w_y (always)	**PIO**/WI

Figure 3.8: Outline of the proof of Lemma 3.5.2. The underlined element in the "$\mathrm{C}_{\mathrm{Mult}}$ knows" column indicates which secret key is used to decrypt information used to construct $[z'']$. For instance, in Game_0, sk_1 is used to obtain P_1 and P'_1, which are used to compute $[z''] = [(P_1 * P'_1)(\omega)]$ by $\mathrm{C}_{\mathrm{Mult}}$. The "or" expressions in the "π''-witness" column specify which π''-witness is used in steps 5.3 and 6 of $\mathrm{C}_{\mathrm{Add}}$ (resp. steps 6.3 and 7 of $\mathrm{C}_{\mathrm{Mult}}$). Hence, in Game_0 the $\mathrm{C}_{\mathrm{Add}}$ circuit uses (sk_1, sk_2) to construct π'' in case $P_1(\omega) = P_2(\omega) = z$ and $P'_1(\omega) = P'_2(\omega) = z'$. Otherwise, $\mathrm{C}_{\mathrm{Add}}$ uses the extracted w_y as witness in π''.

Game_5: Here, once again change the way $\mathrm{C}_{\mathrm{Add}}$ and $\mathrm{C}_{\mathrm{Mult}}$ prepare proofs π''. Specifically, we let $\mathrm{C}_{\mathrm{Add}}$ and $\mathrm{C}_{\mathrm{Mult}}$ to *always* use the hardwired w_y as witness to construct π'', independently of whether or not the encodings h, h' are consistent. Hence, $\mathrm{C}_{\mathrm{Add}}$ and $\mathrm{C}_{\mathrm{Mult}}$ do not need to perform the explicit consistency check anymore. This means that $\mathrm{C}_{\mathrm{Add}}$ no longer needs sk_1 or sk_2, and $\mathrm{C}_{\mathrm{Mult}}$ only needs sk_i (to retrieve P_i and P_i' from c_i and c_i'). These modifications do not change the output *distributions* of $\mathrm{C}_{\mathrm{Add}}$ and $\mathrm{C}_{\mathrm{Mult}}$. Indeed, we have only changed the witness used for π''-proofs. By the *perfect* witness indistinguishability of the proof system (under a hiding CRS), the distributions of the resulting proofs remain identical. Hence, we can use the obfuscator's indistinguishability security against X-IND samplers twice to justify our transition from Game_4 to Game_5.

Observe that in Game_5 the modified public parameters are identically distributed to $\mathcal{PP}_1$. Indeed, we have $y \in \mathcal{L}_{lk}$ by the change introduced in Game_2, the CRS crs' is hiding by the change in Game_4, and circuits $\mathrm{C}_{\mathrm{Add}}$ and $\mathrm{C}_{\mathrm{Mult}}$ always use a hardwired w_y as a witness to construct π''-proofs. Furthermore, $\mathrm{C}_{\mathrm{Mult}}$ uses sk_i to retrieve P_i and P_i', in order to compute $[z''] = [(P_i * P_i')(\omega)]$. These changes render $\mathrm{C}_{\mathrm{Add}}$ identical to $\widehat{\mathrm{C}}_{\mathrm{Add}}$ and $\mathrm{C}_{\mathrm{Mult}}$ identical to $\widehat{\mathrm{C}}^{(i)}_{\mathrm{Mult}}$. □

3.6 Hard Problems

We are now ready to show that the MDDH and RANK problems are hard for our GES. On top of extending AFHLP's results to the graded setting, our results in this section improve [3] in two independent directions: First, we provide simpler and tighter proofs of security. Second, we generalize their results so that for a *single* instantiation of the GES *both* the MDDH and RANK problems are hard.

One corollary of these results is that there are no "zeroizing" attacks on our scheme as such attacks immediately lead to the break of MDDH and RANK problems [30, 33, 51]. Indeed, a second motivation for studying the security of the RANK problem (on top of its applications) is the tighter relation that it has with zeroizing attacks [51, Section 4.4]. Put differently, by proving the RANK problem intractable, we rule out a wider class of zeroizing attacks.

3.6.1 Hardness of MDDH

Recall that the GES of Section 3.4 represents an element $z \in \mathbb{Z}_p$ at level ℓ with polynomials P_1 and P_2 of degree at most $s\ell$ such that $P_j(\omega) = z$, where s is a system parameter. To show that MDDH holds it is sufficient to set $s := 1$, but we prove the following more general result.

Theorem 3.6.1 ($(s\kappa + s - 1)$-SDDH $\implies$ κ-MDDH)**.** *Let Γ be the GES constructed in Section 3.4 with an $s \geq 1$ and with respect to a base group $\mathbb{G}$ and an X-IND-secure probabilistic obfuscator* **PIO**. *Then, for any $\kappa \in \mathbb{N}$ and any* PPT *adversary $\mathcal{A}$ there are* PPT *adversaries $\mathcal{B}_1$, $\mathcal{B}_2$ and $\mathcal{B}_3$ of essentially the same complexity as $\mathcal{A}$ such that for all $\lambda \in \mathbb{N}$*

$$\mathbf{Adv}^{\kappa\text{-mddh}}_{\Gamma,\mathcal{A}}(\lambda) \leq (\kappa+1) \cdot \mathbf{Adv}^{\kappa\text{-switch}}_{\Gamma,\mathcal{B}_1}(\lambda) + \mathbf{Adv}^{}_{\mathbf{PIO},\mathcal{B}_2}(\lambda) + \mathbf{Adv}^{(s\kappa+s-1)\text{-sddh}}_{\mathbb{G},\mathcal{B}_3}(\lambda).$$

Proof. We provide a simpler proof compared to that of [3, Theorem 6.2] at the expense of relying on the slightly stronger κ-SDDH (instead of the $(\kappa-1)$-SDDH) problem for $s = 1$. At a high level, our reduction has two steps: 1) Switch *all* encodings from polynomials of degree 0 to those of degree 1; and 2) Randomize the κ-MDDH challenge using the κ-SDDH instance. The key difference with the proof of [3, Theorem 6.2] is that we no longer need to carry out a two-step process to randomize the exponent of the MDDH challenge. In particular, we do not change the implementation of the multiplication circuit according to a κ-SDDH challenge.

We give a sequence of $\kappa+4$ games, where in the last game, for case $b=1$ the challenge exponent z is also uniformly distributed. Below we let W_i denote the event that Game_i outputs 1.

Game_0: This is the κ-MDDH problem (Figure 2.2, middle). We use $P_{i,1}$ and $P_{i,2}$ to denote the canonical degree-zero representation polynomials of a_i as generated by the sampler $\mathsf{Sam}_1(a_i)$.

Game_1–$\mathrm{Game}_{\kappa+1}$: In this sequence of games, Game_i proceeds similarly to Game_{i-1} with the difference that the representations $P_{i,1}$, $P_{i,2}$ of the i-th challenge encoding h_i (which are at level 1) are no longer of the form

$$P_{i,1}(X) = P_{i,2}(X) := a_i$$

but set to

$$P_{i,1}(X) = P_{i,2}(X) := X^s + a_i - \omega^s.$$

These representation polynomials are valid and of degree *exactly* s, the maximum allowed degree at level 1 with GES parameter s. We claim that

$$|\Pr[\mathrm{W}_{i-1}(\lambda)] - \Pr[\mathrm{W}_i(\lambda)]| \le \mathbf{Adv}^{\kappa\text{-switch}}_{\Gamma,\mathcal{B}_1}(\lambda) \quad \text{for} \quad 1 \le i \le \kappa+1.$$

Given an attacker $\mathcal{A}$ distinguishing Game_{i-1} and Game_i, we build a PPT adversary $\mathcal{B}_1$ against the game κ-Switch of Figure 3.5. Algorithm $\mathcal{B}_1$ outputs $((P_{i-1,1}, P_{i-1,2}), (P_{i,1}, P_{i,2}), \ell = 1, st)$ representing a uniform value a_i in $\mathbb{Z}_p$, where $(P_{i-1,1}, P_{i-1,2})$ is as in Game_{i-1} and $(P_{i,1}, P_{i,2})$ as in Game_i as above. Observe $\mathcal{B}_1$ can indeed construct these polynomials because it knows ω and a_i explicitly (and furthermore they are admissible because at level 1 polynomials can have degree up to s). Algorithm $\mathcal{B}_1$ receives an encoding h_i of a_i that has $(P_{i+b-1,1}, P_{i+b-1,1})$ for a random bit b embedded in it. It uses h_i to simulate Game_{i+b-1} for $\mathcal{A}$, and outputs what $\mathcal{A}$ outputs.

$\mathrm{Game}_{\kappa+2}$: This game only introduces a conceptual change: the i-th source exponent is changed to $a'_i = a_i + \omega^s$ for randomly chosen $a_i \in \mathbb{Z}_p$ and $1 \le i \le \kappa+1$. Also, the polynomial representations of a'_i is set to $P_{\kappa+2,1}(X) \equiv P_{\kappa+2,2}(X) = X^s + a_i$, which has the same degree as the polynomials in $\mathrm{Game}_{\kappa+1}$. This means that the exponent of the target encoding h^*_b when $b=1$ is

$$z^*_1 = Q(\omega) := (\omega^s + a_1) \cdots (\omega^s + a_{\kappa+1}). \tag{3.3}$$

Note that Q has degree $s\kappa+s$ and its $(s\kappa+s)$-th coefficient is 1. The distribution from which the $\kappa+1$ exponents a'_i are drawn has not changed and is uniform. Therefore

$$\Pr[\mathrm{W}_{\kappa+1}(\lambda)] = \Pr[\mathrm{W}_{\kappa+2}(\lambda)].$$

$\mathrm{Game}_{\kappa+3}$: The differences with the previous game are two-fold. First, when $b = 1$, the challenge encoding $h_1^* = [Q(\omega)]$ is generated evaluating polynomial $Q(X)$ at $X = \omega$ in the exponent using $([1], [\omega], \ldots, [\omega^{s\kappa+s}])$, and the explicit knowledge of the coefficients $(q_0, \ldots, q_{s\kappa+s})$ of polynomial $Q(X)$ obtained by expanding Equation (3.3). This change is purely conceptual.

The second difference is that we obfuscate circuit $\mathrm{C}^*_{\mathrm{Mult}}$ which has the powers $([1], [\omega], \ldots, [\omega^{s\kappa}])$ hardwired in and computes the map implicitly in the exponent. In more detail, this circuit extracts the representation polynomials P_1, P_1' from the input encodings (at levels ℓ and ℓ' respectively) and evaluates $P'' := P_1 * P_1'$ at ω in the exponent using $([1], [\omega], \ldots, [\omega^{s\kappa}])$. The latter is possible because by the perfect soundness of the proof system under a binding CRS, P_1 (respectively, P_1') is of degree at most $s\ell$ (respectively, $s\ell'$), and therefore P'' is of degree at most $s(\ell+\ell') \leq s\kappa$. This modification therefore results in a functionally equivalent circuit (both compute $[P''(\omega)]$). Since $\mathrm{C}^*_{\mathrm{Mult}}$ is of polynomial size, we conclude that obfuscations of these two circuits are indistinguishable:

$$|\Pr[\mathrm{W}_{\kappa+1}(\lambda)] - \Pr[\mathrm{W}_{\kappa+2}(\lambda)]| \leq \mathbf{Adv}_{\mathbf{PIO},\mathcal{B}_2}(\lambda).$$

$\mathrm{Game}_{\kappa+4}$: We regard the degree $(s\kappa + s)$ polynomial $Q(X)$ of Equation (3.3) as a multivariate $\mathbb{Z}_p$-polynomial $Q'(Y_1, \ldots, Y_{s\kappa+s})$ in $s\kappa+s$ unknowns by renaming variables X^i to Y_i. In this game when $b = 1$ the challenger samples random $\omega, \tau \in \mathbb{Z}_p$ and sets

$$h_1^* = [z_1^*] := [Q'(\omega, \omega^2, \ldots \omega^{s\kappa+s-1}, \tau)],$$

where Q' is evaluated in the exponent using $([\omega^i])_{0\leq i\leq s\kappa+s-1}$ and $[\tau]$. We emphasize that circuit $\mathrm{C}^*_{\mathrm{Mult}}$ still has $([1], [\omega], \ldots, [\omega^{s\kappa}])$ hardwired as in the previous game. We claim that

$$|\Pr[\mathrm{W}_{\kappa+3}(\lambda)] - \Pr[\mathrm{W}_{\kappa+4}(\lambda)]| \leq \mathbf{Adv}^{(s\kappa+s-1)\text{-sddh}}_{\mathbb{G},\mathcal{B}_3}(\lambda).$$

This immediately follows because an adversary $\mathcal{B}_3$ against $(s\kappa + s - 1)$-SDDH on receiving challenge $(([\omega^i])_{0\leq i\leq s\kappa+s-1}, [\tau])$ can simulate $\mathrm{Game}_{\kappa+3}$ if $\tau = \omega^{s\kappa+s}$, or $\mathrm{Game}_{\kappa+4}$ if τ is random.

To see that $\Pr[\mathrm{W}_{\kappa+4}] = 1/2$ it suffices to show that in $\mathrm{Game}_{\kappa+4}$ exponent z_1^* is randomly distributed over $\mathbb{Z}_p$. This follows because the leading coefficient of Q' is 1, and therefore the map $f(X) := Q(\omega, \ldots, \omega^{s\kappa+s-1}, X)$ defines a bijection over $\mathbb{Z}_p$ mapping a uniform τ into a uniform $z_1^* = f(\tau)$. □

3.6.2 Downgrading Attacks

It might appear that our GES could be subject to a "downgrading" attack as follows. Start with any consistent encoding h at level ℓ whose representation polynomial is of degree 0. Then "maul" h into an encoding at a lower level $\ell' < \ell$ by simply changing ℓ to ℓ' in h. Then use this malleability to attack, say, MDDH where challenge encodings are canonical and of degree 0 (see Section 3.4.5).

What is crucial and prevents this downgrade attack is the proof system. The consistency proof π proves that the encrypted values correspond to a polynomial P

of degree up to ℓ such that $P(\omega) = z$. Note that this statement *depends on* ℓ. Hence, a proof for a level-2 encoding cannot be "reused" for a level-1 encoding, as in the attack: a single proof will not necessarily pass against two different statements even if they both have the same witness. In order to downgrade, the proof would have to be changed.

Indeed, suppose that one had a method for changing a proof π_2 of a level-2 encoding to a proof π_1 of the level-1 encoding (that is derived by simply omitting encrypted coefficients, as in a downgrading attack). Consider what happens if one start with equivalent level-2 encoding (in the sense of our switching lemma) with degree-2 polynomials P. Then, the statement that π_1 proves becomes false, so any such attack would contradict the soundness of the proof system.

3.6.3 Hardness of RANK

We show that the RANK problem for the construction with $s \geq 2$ (where encodings at level ℓ have representation polynomials of degree at most $s\ell$) is hard. A standard hybrid argument shows that the hardness of $(\kappa, m, n, r, r+1, l)$-RANK implies the hardness of $(\kappa, m, n, r_0, r_1, l)$-RANK for all $r_1 > r_0$. We prove that the former problem is hard next.

Theorem 3.6.2 (SDDH $\implies$ RANK)**.** *Let Γ be the GES constructed in Section 3.4 with an $s \geq 2$ and with respect to a base group $\mathbb{G}$ and an X-IND-secure probabilistic obfuscator* **PIO**. *Let $p(\lambda)$ denote the size of the base group $\mathbb{G}$, (κ, m, n, r) be integers with $r \geq \kappa$, and $l : [m] \times [n] \longrightarrow [\kappa]$ be an arbitrary level function. Then, for any* PPT *adversary $\mathcal{A}$ there are* PPT *adversaries $\mathcal{B}_1$, $\mathcal{B}_2$ and $\mathcal{B}_3$ of essentially the same complexity as $\mathcal{A}$ such that for all $\lambda \in \mathbb{N}$*

$$\mathbf{Adv}^{(\kappa,m,n,r,r+1,l)\text{-rank}}_{\Gamma,\mathcal{A}}(\lambda) \leq 3 \cdot \mathbf{Adv}^{\kappa\text{-switch}}_{\Gamma,\mathcal{B}_1}(\lambda) + \mathbf{Adv}_{\mathbf{PIO},\mathcal{B}_2}(\lambda) + (s\kappa - 1) \cdot \mathbf{Adv}^{(s\kappa-1)\text{-sddh}}_{\mathbb{G},\mathcal{B}_3}(\lambda) + \frac{1}{p(\lambda)}.$$

The proof technique follows that of AFHLP [3, Theorem 7.1], and is adapted to the graded setting where the entries of $\overline{\mathbf{H}}_b$ may lie at *any* (nonzero) level. Here we also present a tighter reduction and prove the theorem more generally for any $s \geq 2$. We start with an outline of the main idea and give a formal proof afterwards.

Outline. To prove the theorem we need to generate a matrix $\overline{\mathbf{H}}_0$, which encodes a rank-r matrix $\mathbf{M}_0$ with entries at arbitrary levels, and gradually transform it into a matrix $\overline{\mathbf{H}}_1$ that encodes a rank-$(r+1)$ matrix $\mathbf{M}_1$.[8]

EMBEDDING THE SDDH CHALLENGE. Ideally, we would like to reduce RANK to the weakest 1-SDDH assumption. To this end, consider the $\mathbb{G}$-matrix

$$[\mathbf{W}] := \begin{bmatrix} [1] & [\omega] \\ [\omega] & [\tau] \end{bmatrix},$$

where $\omega \in \mathbb{Z}_p$ is the secret exponent of Γ. This matrix can be formed (in the exponent) from a 1-SDDH challenge in $\mathbb{G}$, and we will exploit the fact that if $\tau = \omega^2$ then

[8] Compare this with the MDDH case, where the challenge lives in the last level $S_\kappa = \mathbb{G}$.

$\mathbf{W}$ has rank 1, and if τ is uniform then it has rank 2 (with overwhelming probability). We then rely on the observation that an encoding of ω^2 can be obtained at any level knowing $[\omega^2]$ *only in the exponent*. Concretely, the polynomial representation of ω^2 at an arbitrary level $\ell \geq 1$ can be set to $P(X) = X^2$ as we allow polynomials representations of degree at most $s\ell$. Here is where we use the fact that $s \geq 2$ as this technique would not work for $s = 1$.

LIFTING. To generate an $m \times n$ matrix $\mathbf{M}_0$ of rank r (or $r+1$ after the 1-SDDH hop above) we use the standard technique of embedding the identity matrix $\mathbf{I}[r-1]$ in the diagonal:

$$\mathbf{M}_0 = \begin{bmatrix} \mathbf{W} & & \\ & \mathbf{I}_{r-1} & \\ & & \mathbf{0} \end{bmatrix},$$

where $\mathbf{0}$ denotes a zero matrix of dimension $(m-r-1) \times (n-r-1)$ to bring the matrix up to dimension $m \times n$. Moreover, via the random self-reducibility of the RANK problem (via multiplication of the instance with explicit random invertible matrices) matrix $\mathbf{M}_0$ can be randomized to a uniform one in $\mathrm{Rk}_r(\mathbb{Z}_p^{m \times n})$. Since this randomization is *linear*, we can generate $\overline{\mathbf{H}}_0$ with entries at arbitrary levels even when ω and τ are only known in the exponent.

BREAKING CORRELATION WITH $\mathrm{C}_{\mathrm{Mult}}$. We can construct a circuit that is functionally equivalent to $\mathrm{C}_{\mathrm{Mult}}$ and that uses the powers $[\vec{\omega}] = ([1], [\omega], \ldots, [\omega^{s\kappa}])$ only. The latter circuit outputs $([z], c_1, c_2, \pi, \ell)$ (or simply $[z]$ for level κ), where z is the evaluation of a polynomial of degree at most $s\kappa$ at point ω. We then invoke the q-SDDH assumptions for $2 \leq q \leq s\kappa - 1$ to gradually randomize the higher powers $[\omega^i]$ for $i \geq 2$ one at a time, so that we can embed a 1-SDDH tuple in the challenge matrix $\overline{\mathbf{H}}_0$ as explained above. This yields the encoding of a rank-$(r+1)$ matrix. □

Proof. We give a sequence of games starting with the RANK game, and finishing with a game that samples matrices of rank $r+1$, independently of the bit b, with overwhelming probability. The games are virtually the same as those provided in [3], however we have saved some games reducing the complexity of the reduction (by a factor of $mn - 3$ in the advantage of game κ-Switch.) Below, we let W_i denote the event that Game_i outputs 1.

Game_0**:** This is the original $(\kappa, m, n, r, r+1, l)$-RANK problem for $r \geq \kappa$ as shown in Figure 2.2 (right).

Game_1**:** When $b = 0$, this game encodes the random matrix $\mathbf{M}_0$ (over $\mathbb{Z}_p$) of rank r differently. Instead of using constant polynomials $m_{1,2}$, $m_{2,1}$, and $m_{2,2}$ for entries $(1,2)$, $(2,1)$ and $(2,2)$ of $\mathbf{M}_0$, it uses the following polynomial representations to prepare the encoding matrix, where ω is the secret exponent of Γ.

$$\begin{aligned} P_1(X) = P_2(X) &:= X + m_{1,2} - \omega && \text{for entry } (1,2) \\ P_1(X) = P_2(X) &:= X + m_{2,1} - \omega && \text{for entry } (2,1) \\ P_1(X) = P_2(X) &:= X^2 + m_{2,2} - \omega^2 && \text{for entry } (2,2) \end{aligned}$$

These encodings are of degrees exactly 1, 1 and 2 respectively. Hence, using an argument similar to that for Theorem 3.6.1 we get that for an algorithm $\mathcal{B}_1$

$$|\Pr[\mathrm{W}_0(\lambda)] - \Pr[\mathrm{W}_1(\lambda)]| \leq 3 \cdot \mathbf{Adv}^{\kappa\text{-switch}}_{\Gamma,\mathcal{B}_1}(\lambda).$$

Game_2**:** In this game, when $b = 0$ we change the way the matrix $\overline{\mathbf{H}} = \overline{\mathbf{H}}_0$ is generated. Define the $\mathbb{Z}_p$-matrix

$$\mathbf{W} := \begin{bmatrix} 1 & \omega \\ \omega & \omega^2 \end{bmatrix},$$

for secret exponent $\omega \in \mathbb{Z}_p$, and the $(m \times n)$ matrix $\mathbf{M}'$ over $\mathbb{Z}_p$

$$\mathbf{M}' := \begin{bmatrix} \mathbf{W} & & \\ & \mathbf{I}_{r-1} & \\ & & \mathbf{0} \end{bmatrix}, \tag{3.4}$$

where $\mathbf{I}[r-1]$ is the identity matrix of order $r-1$, and $\mathbf{0}$ is the $(m-r-1) \times (n-r-1)$ zero matrix.

The game uses $[\mathbf{W}]$, the matrix $\mathbf{W}$ in the exponent with entries in $\mathbb{G}$, to form an encoded matrix $\overline{\mathbf{H}}'$ corresponding to $\mathbf{M}'$. We note that the polynomial representations used to encode entries $m'_{1,2} = \omega$ and $m'_{2,1} = \omega$ are $P_1(X) = P_2(X) := X$, and that used to encode entry $m'_{2,2} = \omega^2$ is $P_1(X) = P_2(X) := X^2$ (as in Game_1). Finally, the game sets $\overline{\mathbf{H}} = \mathbf{L} \cdot \overline{\mathbf{H}}'$, where $\mathbf{L}$ is a random invertible matrix of dimension $m \times m$ over $\mathbb{Z}_p$. Observe that this transformation is linear, and therefore the game can construct the entries of $\overline{\mathbf{H}}$ from the encoded matrix $\overline{\mathbf{H}}'$ and the $\mathbb{Z}_p$-matrix $\mathbf{L}$ using algorithm Add of Γ.

We claim that

$$\Pr[\mathrm{W}_1(\lambda)] = \Pr[\mathrm{W}_2(\lambda)].$$

To see this, we first note that challenge matrix $\overline{\mathbf{H}}$ is an encoding of matrix $\mathbf{M} := \mathbf{L} \cdot \mathbf{M}'$ (by correctness of Add). Next, to see that $\mathbf{M}$ is randomly distributed over $m \times n$ matrices of rank r, observe that: (1) $\mathbf{M}'$ has rank r because $\mathbf{W}$ has rank 1 and (2) $\mathbf{M} = \mathbf{L} \cdot \mathbf{M}'$ is a uniformly distributed rank-r matrix, because the left action of invertible matrices $\mathrm{GL}_{m\times m}(\mathbb{Z}_p)$ is transitive in the set of $\mathbb{Z}_p$-matrices of dimension $m \times n$ and rank r (cf. the random self-reducibility of the RANK problem).

Game_3**:** The difference with the previous game is that Setup outputs (the obfuscation of) a different circuit $\mathrm{C}^*_{\mathrm{Mult}}$ that has the tuple $[\vec{\omega}_s] = ([1], [\omega], \ldots [\omega^{s\kappa}])$ hardwired in, extracts polynomials P_1 and P'_1 from its inputs, and evaluates $P_1 * P'_1$ at point ω in the exponent using $[\vec{\omega}_s]$. We claim that for an algorithm $\mathcal{B}_2$

$$|\Pr[\mathrm{W}_2(\lambda)] - \Pr[\mathrm{W}_3(\lambda)]| \leq \mathbf{Adv}_{\mathbf{PIO},\mathcal{B}_2}(\lambda).$$

This follows from the fact that $\mathrm{C}^*_{\mathrm{Mult}}$ knowing $[\vec{\omega}_s]$ is functionally equivalent to $\mathrm{C}_{\mathrm{Mult}}$ knowing ω in the clear. Indeed the perfect soundness and completeness of the NIZK proof system (with respect to a binding CRS) guarantees that the extracted polynomial representations P_1 and P'_1 are of degrees ℓ and ℓ' such

that $\ell + \ell' \leq 2\kappa$. Hence, the powers of ω included in $[\vec{\omega}_s]$ suffice to evaluate $P_1 * P_1'$ in the exponent, and explicit knowledge of powers of ω are no longer needed in the subsequent games.

Game_{3+i} **for** $1 \leq i \leq s\kappa - 2$**:** In these games we randomize the powers of ω one by one, starting with the highest power. Here Setup instead of $[\vec{\omega}_s]$ includes the following vector $[\vec{g}_i]$ in gpk.

$$[\vec{g}_i] := ([1], [\omega], \ldots, [\omega^{s\kappa-i}], [\tau_{s\kappa-i+1}], [\tau_{s\kappa-i+2}], \ldots, [\tau_{s\kappa}]),$$

where $\tau_{s\kappa-i+j}$ are fresh random values in $\mathbb{Z}_p$. Observe that in particular (the obfuscation of) circuit $\mathrm{C}^*_{\mathrm{Mult}}$ has $[\vec{g}_i]$ hard-coded and therefore is not a bilinear map anymore. An attacker against $(s\kappa-i)$-SDDH can embed a challenge tuple $([1], [\omega], \ldots, [\omega^{s\kappa-i}], [\tau_{s\kappa-i+1}])$ in the first $s\kappa - i + 1$ positions of $[\vec{g}_i]$. Then, if $\tau_{s\kappa-i+1} = \omega^{s\kappa-i+1}$ this simulates Game_{3+i-1}, otherwise it simulates Game_{3+i}. This shows that for $1 \leq i \leq s\kappa - 2$:

$$|\Pr[\mathrm{W}_{3+i-1}(\lambda)] - \Pr[\mathrm{W}_{3+i}(\lambda)]| \leq \mathbf{Adv}^{(s\kappa-i)\text{-sddh}}_{\mathbb{G},\mathcal{B}_3'}(\lambda).$$

$\mathrm{Game}_{s\kappa+2}$**:** We continue the above sequences of game with $i = s\kappa - 1$ but now also change how $\mathbf{W}$ is generated. The game samples random ω and τ_i for $2 \leq i \leq s\kappa$, in $\mathbb{Z}_p$. Then it hard-codes $[\vec{g}_{s\kappa-1}] = ([1], [\omega], [\tau_2] \ldots, [\tau_{s\kappa}])$ in $\mathrm{C}^*_{\mathrm{Mult}}$, and sets the minor $\mathbf{W}$ of $\mathbf{M}'$ (see Equation (3.4)) to

$$\mathbf{W} = \begin{bmatrix} 1 & \omega \\ \omega & \tau_2 \end{bmatrix}.$$

An encoding of ω and τ_2 can be generated only with the knowledge of $[\omega]$, $[\tau_2]$ (setting the polynomial representations to $P_i^{(\omega)}(X) = X$, and $P_i^{(\tau)}(X) = X^2$ respectively). An adversary against 1-SDDH can embed its challenge $([\omega], [\tau_2])$ to simulate $\mathrm{Game}_{s\kappa+1}$ if $\tau_2 = \omega^2$, or $\mathrm{Game}_{s\kappa+2}$ if τ_2 is random. Thus

$$|\Pr[\mathrm{W}_{s\kappa+1}(\lambda)] - \Pr[\mathrm{W}_{s\kappa+2}(\lambda)]| \leq \mathbf{Adv}^{1\text{-sddh}}_{\mathbb{G},\mathcal{B}_3'}(\lambda).$$

Finally, $\Pr[\mathrm{W}_{s\kappa+2}(\lambda)] \leq 1/2 + 1/p(\lambda)$ because even when $b = 0$, the rank of $\mathbf{M} = \mathbf{L} \cdot \mathbf{M}'$ is $r+1$ with overwhelming probability over the choice of τ_2. (Concretely, the probability that $\tau_2 = \omega^2$ is $1/p(\lambda)$.) □

The simplified advantage term in the theorem statement follows form the observation that for any $q' \leq q$ and any algorithm $\mathcal{B}_3'$, there is an algorithm $\mathcal{B}_3$ of essentially the same complexity such that

$$\mathbf{Adv}^{q'\text{-sddh}}_{\mathbb{G},\mathcal{B}_3'}(\lambda) \leq \mathbf{Adv}^{q\text{-sddh}}_{\mathbb{G},\mathcal{B}_3}(\lambda).$$

This can be proved easily using a re-randomization of the generator for the base group.

Chapter 4

Compact Lossy Trapdoor Functions from Multilinear Maps

In this chapter, we contribute to the question about how multilinear maps or graded encoding schemes, respectively, can be used for cryptographic purposes. Namely, we explain how both tools can be used to construct compact lossy trapdoor functions in the DLOG-based setting.

4.1 Overview

LOSSY TRAPDOOR FUNCTIONS (LTDFS). A family of lossy trapdoor functions is a collection $\{f\}$ of functions $f : \{0,1\}^n \to \{0,1\}^*$, some of which are invertible (given a suitable trapdoor f^{-1}), and some of which are lossy (in the sense that their image is smaller than their preimage). Moreover, invertible and lossy functions should be computationally indistinguishable.

Introduced by [99], LTDFs have proven to be an extremely useful cryptographic building block. For instance, LTDFs imply various types of public-key encryption (including IND-CCA secure [99, 94], deterministic [8], and selective-opening secure [7] encryption), oblivious transfer protocols [99], and several other fundamental building blocks (including collision-resistant hash functions [99], trapdoor one-way functions [99], and correlation-secure trapdoor one-way functions [103]). As can be expected for such a foundational primitive, there exist various constructions of LTDFs from different computational assumptions (e.g., from DDH [99, 115], k-Linear [46, 23, 115], LWE [99, 115], DCR [46, 67], QR [46], or specialized assumptions [68, 115]).

However, perhaps somewhat surprisingly, all known LTDF instances in the discrete log setting (i.e., from the DDH, the k-Linear, or similar assumptions in cyclic groups of known order) share one disadvantage. Namely, the size of the public description of the function as well as its output size is at least a linear number of group elements in the (bit)size n of the function's input.[1] In other words, we know of no *compact* LTDFs in the discrete log setting.[2]

OUR CONTRIBUTION: LTDFS WITH COMPACT PUBLIC DESCRIPTION. In this work, we improve the situation by constructing LTDFs with at least a compact public description size. Before, known constructions of LTDFs in the discrete log setting with input

[1] Perhaps this may appear less surprising, given that we do not even know a mere trapdoor one-way function in the discrete log setting with a more compact description or output.

[2] Note that straightforward generic approaches (such as using a function with small input size several times, on different parts of a larger input, and concatenating the results) may extend the input size at the cost of lossiness. We are not interested in such trade offs here.

(bit)size n have a public description size of $\mathbf{O}(n^2)$ group elements [99, 46, 115], or – when assuming a bilinear map – $\mathbf{O}(n)$ group elements [23].

We describe two new LTDF constructions, both in the discrete log setting, and assuming that a multilinear map is available. Our first construction assumes an asymmetric k-linear map, in the sense that all input elements come from different groups. Our second construction assumes a symmetric k-linear map, in the sense that all input elements for the map come from the same group. Both of our LTDFs have a very short, $\mathbf{O}(k \cdot n^{2/k})$-sized (resp. $\mathbf{O}(n^{2/k})$-sized) public description in the respective source group(s) of the multilinear map. Upon invocation, this short description is then uncompressed into an $\mathbf{O}(n^2)$-sized public description for the LTDF from [99] (resp. [46, 115]) in the target group. Then the LTDF from [99] (resp. [46, 115]) is applied with the uncompressed public description.

This is conceptually similar to the way [23] first uncompress a public description into the target group of a bilinear map, and then apply the LTDF from [99]. However, even though it yields similarly efficient LTDFs in the bilinear case, our compression strategy is very different from that of [23]. In particular, as we will explain below, while the strategy of [23] does not scale beyond the bilinear case, ours does.

4.1.1 More Technical Details

THE LTDF OF [99]. For a more detailed technical overview, we focus on our first LTDF. Our LTDF builds upon the LTDF of [99] (the "PW-LTDF"), which we now briefly review. The PW-LTDF works in a cyclic group $G = \langle g \rangle$ of prime order p, and its public description consists of a matrix $C = \mathsf{Enc}(\mathbf{M}) = (c_{i,j})_{i,j=1}^{n}$ that is the component-wise encryption of a matrix $\mathbf{M} := (m_{i,j}) \in \mathbb{Z}_p^{n \times n}$.[3] The underlying encryption scheme must be additively homomorphic, so that an evaluation $f(\vec{x}) = \mathsf{Enc}(\vec{x} \cdot \mathbf{M}) = (\prod_{i=1}^{n} c_{i,j}^{x_i})_{j=1}^{n}$ (which is simply a component-wise encryption of the vector $\vec{x} \cdot \mathbf{M}$) can be computed efficiently from C and $\vec{x} = (x_i)_{i=1}^{n} \in \{0,1\}^n$. If $\mathbf{M}$ is the all-zero matrix, then this evaluation becomes lossy, and if $\mathbf{M}$ is an invertible matrix, then also F can be efficiently inverted (e.g., by decrypting and then applying $\mathbf{M}^{-1}$)[4]. Thus, when implemented with ElGamal encryption, lossy and invertible LTDF instances are computationally indistinguishable under the DDH assumption in G.

THE STRATEGY OF BOYEN AND WATERS TO COMPRESS THE PUBLIC DESCRIPTION OF THE PW-LTDF. Clearly, the public description size of $\mathbf{O}(n^2)$ group elements of the PW-LTDF is unsatisfactory. Hence, [23] devise a strategy to compress the public description of the PW-LTDF in a setting with a bilinear map $e : G \times G \to G_T$. The overall idea is to construct a matrix C of encryptions as above in the target group G_T, and then proceed as in the PW-LTDF. Specifically, [23] find a way to suitably blind group elements $u_i, v_i \in G$ (for $i \in \{1, \ldots, n\}$), such that encryptions of $e(u_i, v_j)$ can be computed exactly for $i \neq j$. This yields all elements of a ciphertext matrix C (of zero-encryptions) as with the PW-LTDF, except for the diagonal elements. Depending on how the diagonal of C is completed, the corresponding PW-LTDF will be

[3] For now, we ignore here a subtlety concerning the used encryption public keys and choice of random coins.

[4] In [99], $\mathbf{M}$ is always the identity matrix in case of an invertible function.

lossy or injective. Hence, depending on whether a lossy or injective function is desired, the corresponding diagonal elements are added (uncompressed) to the public description. This leads to a public description size of $\mathbf{O}(n)$ group elements (where the diagonal and non-diagonal elements of C are both encoded with $\mathbf{O}(n)$ group elements). However, note that while it may be possible to further compress the description of the non-diagonal elements of C in a multilinear setting, it is unclear how the diagonal of C itself can be compressed at all.

OUR STRATEGY. Our LTDF assumes an asymmetric multilinear map, i.e., we assume cyclic groups $G_1, \ldots, G_k, G_T$ and a k-linear map $e : G_1 \times \cdots \times G_k \to G_T$. We will also assume that the DDH assumption holds in each group G_i. Like [23], we construct a PW-LTDF description in G_T from information in the source groups. To this end, we start with many small matrices $\mathbf{M}_i \in \mathbb{Z}_p^{\sqrt[k]{n} \times \sqrt[k]{n}}$ (for $i \in \{1, \ldots, k\}$), and their respective component-wise encryptions $C_i = \mathsf{Enc}(\mathbf{M}_i)$ in G_i. We then compute C from the C_i as

$$C := C_1 \circ \cdots \circ C_k \tag{4.1}$$

for a suitable operation $\circ$ on encrypted matrices. Ideally, we would like to have that $\mathsf{Enc}(\mathbf{M}_i) \circ \mathsf{Enc}(\mathbf{M}_j) = \mathsf{Enc}(\mathbf{M}_i \otimes \mathbf{M}_j)$ for the Kronecker product $\otimes$. Namely, such a map would multiply the matrix dimensions, which means that a sufficiently large matrix C can be constructed from few and small C_i. (In other words, our compression strategy would lead to small public descriptions.) Besides, $\mathrm{rank}(\mathbf{M}_i \otimes \mathbf{M}_j) = \mathrm{rank}(\mathbf{M}_i) \cdot \mathrm{rank}(\mathbf{M}_j)$, so we end up with $C = \mathsf{Enc}(\mathbf{M})$ with all-zero, resp. invertible $\mathbf{M}$ whenever all $\mathbf{M}_i$ are all-zero, resp. invertible. Hence, depending on the $\mathbf{M}_i$, the eventual LTDF becomes lossy or invertible. (Assuming a secure encryption scheme, these cases are computationally indistinguishable.)

WHY STRAIGHTFORWARD APPROACHES FAIL. Unfortunately, we do not know how to efficiently implement an operation $\circ$ as above. Intuitively, the reason is that there appears to be no suitable encryption scheme that would allow to *multiply* encrypted messages using a multilinear map. To explain the issues that arise, let us consider two natural candidates of suitable encryption schemes. Perhaps the most straightforward idea would be to use ElGamal "with the message in the exponent" (as used in [99]), such that the encryption of m under public key $pk = [z]$ and with random coins r is $\mathsf{Enc}_{pk}(m; r) = ([r], pk^r \cdot [m])$. To implement an operation $\otimes$ on the plaintext space publicly (i.e., with knowledge only of the C_i), we would need to be able to multiply ciphertexts under different public keys, possibly using a multilinear map. Concretely, given $\mathsf{Enc}_{pk}(m; r) = ([r], [zr + m])$ and $\mathsf{Enc}_{pk'}(m'; r') = ([r'], [z'r' + m'])$, we would need to compute $pk'' = [r'']$ and $\mathsf{Enc}_{pk''}(mm'; r'') = ([r''], [z''r'' + mm'])$ for some r'', z''. While we *can* compute $[rr']_T$ using a bilinear map (implicitly setting $r'' := rr'$), it is unclear how one would compute $[z''r'' + mm']_T$ along with a suitable $pk'' = [z'']_T$. (For instance, we could compute $[(zr + m)(z'r' + m')]_T = [zz'rr' + mm' + zrm' + z'r'm]_T$, which equals our desired result $[zz'rr' + mm']_T$ except for the unwanted "cross-terms" $zrm' + z'r'm$. Unfortunately, we do not know how to isolate these terms.)

Another approach would involve the Boneh-Goh-Nissim PKE scheme [13]. This scheme works in a cyclic group of composite order $N = pq$, and ciphertexts are of the form $c := g_p^r \cdot g_q^m$ for public elements g_p, g_q of order p, resp. q, and random coins r. Note that in case of such groups, a multilinear map must satisfy $e(g_p, g_q) = 1$, so

that plaintexts can be easily multiplied using $e(g_p^r g_q^m, g_p^{r'} g_q^{m'}) = g_p^{rr'} g_q^{mm'}$. The problem with this approach is hence not the operation $\circ$ itself, but establishing that lossy and invertible C_i are computationally indistinguishable. At this point, we should mention one necessary property of the PW-LTDF encrypted matrix C we have neglected so far. Namely, the used public keys and encryption random coins in C must be correlated (or, re-used), in a way such that the encryption random coins of $f(\vec{x}) = \mathsf{Enc}(\mathbf{M} \cdot \vec{x})$ reveal only a limited amount of information about $\vec{x}$. With the BGN scheme, this translates to the property that the matrix $\mathbf{R} = (r_{i,j})_{i,j=1}^n$ of random coins used to produce $C = (\mathsf{Enc}(m_{i,j}; r_{i,j}))_{i,j}$ has low rank. With an operation $\circ$ sketched above, this translates to the requirement that the random matrices $\mathbf{R}_i$ used to produce the building block matrices C_i are similarly of low rank. With this requirements on the random coins, it seems hard to show that C_i that encrypt low-rank $\mathbf{M}_i$ are computationally indistinguishable from C_i that encrypt invertible $\mathbf{M}_i$.[5]

OUR APPROACH. In a nutshell, our solution is to use the ElGamal encryption scheme (as described above), and to implement an operation $\circ$ as in Equation (4.1) that only implements an "approximation" of the Kronecker product on the plaintext matrices. Concretely, we start with ElGamal ciphertext matrices $C_i = \mathsf{Enc}(\mathbf{M}_i)$ as above. We then define $C_i \circ C_j$ as the Kronecker product of matrices, where multiplication of encryptions is defined as component-wise pairing.[6] Note that this way, the matrices $\mathbf{R}_i$ of random coins are combined through a Kronecker product (such that the final randomness matrix $\mathbf{R}$ of the final description C from Equation (4.1) is of low rank if all $\mathbf{R}_i$ are low-rank), but the plaintext matrix $\mathbf{M}$ is "polluted" (even recursively) with cross-terms of the form $z'r'm + rzm'$.

The good news with this is that it is easy to see that encryptions C_i of all-zero matrices lead to an encryption C of an all-zero matrix, suitable to evaluate a lossy function in the sense of [99]. The bad news, however, is that it is not immediately clear how an encryption C of an invertible matrix can be generated.[7] Hence, the main technical work is to show that C encrypts an invertible matrix (at least with high probability) once all C_i encrypt invertible matrices. (The necessary argument proceeds inductively and uses nontrivial linear algebra.)

OUR LTDF IN THE SYMMETRIC SETTING. The LTDF sketched above uses the ElGamal encryption scheme, whose security relies on the DDH assumption. Hence, we cannot use that assumption in a setting with a symmetric multilinear map (i.e., with a multilinear map $e : G \times \cdots \times G \to G_T$). Instead, in a symmetric setting, we offer a construction based on the k-Linear-based LTDF from [46, 115]. Very briefly, this LTDF uses as public description a matrix $C = [\mathbf{M}] = ([m_{i,j}])_{i \in \{1,\ldots,n\}}$ of group ele-

[5] Of course, one can always assume that such an indistinguishability holds, which essentially implies that we assume indistinguishability of lossy and invertible functions. However, we could not show that this is implied by a more "natural" computational assumption.

[6] More formally, for $C = (c_{i,j})_{i,j=1}^m$ and $C' = (c'_{i,j})_{i,j=1}^{m'}$, we let $C \circ C'$ be the ciphertext matrix $C'' = (c''_{i,j})_{i,j=1}^{m \cdot m'}$ with $c''_{(i-1)m'+i'+1,(j-1)m'+j'+1} = c_{i,j} \circ c'_{i',j'}$, where multiplication of individual ciphertexts $c \circ c' = (g^r, g^{zr+m}) \circ (g^{r'}, g^{z'r'+m'}) = (e(g,g)^{rr'}, e(g,g)^{zz'rr'+mm'+z'r'm+rzm'})$ is defined through component-wise pairing. We implicitly set the public key of $c \circ c'$ as $e(g,g)^{zz'}$.

[7] While one would expect that C encrypts a "reasonably random" matrix once all C_i encrypt random matrices, note that this is not true information-theoretically: the combined entropy contained in all C_i is much lower than the entropy of a uniformly random C.

ments, where the function becomes invertible iff the exponent matrix $\mathbf{M} = (m_{i,j})$ has full rank. Using a Kronecker product strategy similar to (but simpler as) the one above, we can reconstruct $C \in G_T^{n \times n}$ as a Kronecker product "in the exponent" $C = C' \circ \cdots \circ C'$ for $C' \in G^{\ell \times \ell}$.

The advantage over our asymmetric construction is a smaller description size (since only one building block matrix has to be stored). The disadvantage is that we have to rely on the k-Linear assumption (which does not hold, e.g., in the setting of the recent approximative multilinear map candidate from [51]). In particular, already the matrix C' has to be of dimension $\ell > k$ in a setting with a k-linear map. Furthermore, as inherited from the k-Linear-based LTDFs from [46, 115], the lossiness of our LTDF is lower than in the asymmetric case.

POSSIBLE IMPLEMENTATIONS OF MULTILINEAR MAPS. We formulate our LTDFs with generic multilinear maps. In the bilinear case, this means they can be implemented using standard (symmetric or asymmetric) pairings. In the multilinear case (with $k > 2$), one could hope that our constructions can be implemented with the recent approximative multilinear map candidates due to [51], resp. [35]. Unfortunately, this does not appear to be the case, even regardless to any security considerations: due to the randomized nature of encodings of group elements in [51, 35], the LTDF's output may come with a small noise, which may depend on the input even in the lossy case.[8]

The recently proposed approximation of a multilinear map of [3] (see 3.3 for a summary), on the other hand, seems to be suitable for implementing our LTDFs. Like [51, 35], their construction also makes use of randomized encodings in the source group, but maps them to unique encodings. The output of the map is in fact an element of a cyclic group. This allows further computations with output elements of the multilinear map as needed, e.g., for evaluating and inverting our LTDFs. Furthermore, the same techniques as in [99, 23, 46] can be used to argue why our LTDFs are lossy.

4.1.2 Efficiency

To construct an $n \times n$-matrix in a setting with an asymmetric (resp. symmetric) k-linear map, we require k (resp. 1) $\sqrt[k]{n} \times \sqrt[k]{n}$-matrices. (Note that in case of a symmetric multilinear map, we must have $\sqrt[k]{n} > k$ to enjoy any kind of lossiness.) This results in a public description of $2kn^{2/k}$ (resp. $n^{2/k}$) G-elements.

The computational complexity of a function evaluation mainly depends on the time to uncompress the compressed public description. In the asymmetric (resp. symmetric) setting, this amounts to $2n^2$ (resp. n^2) evaluations of the k-linear map. Additionally, there are $2n^3$ (resp. n^3) group operations in order to actually evaluate the function.

In Table 4.1, we contrast and compare these figures to that of existing LTDFs in the discrete log setting.

[8] One cannot easily use the noiseless "canonical representation" of an encoding from [51, 35], since these canonical representation do not allow further computations (as necessary for LTDF inversion). Alternatively, one could hope to argue that even the noise information could be made lossy (as, e.g., in [99] for the LWE-based LTDF). However, due to the different nature of noise in [51, 35], it does not seem clear how to do so.

Scheme	Assumption	Type of map	Description size	Lossiness
PW08 [99]	DDH	none	$n^2 + 2n$	$n - \log_2(\lvert G\rvert)$
FGKRS10 [46, 115]	k-LIN	none	n^2	$n - k\log_2(\lvert G\rvert)$
BW10 [23]	bilinear DDH	bilinear (A/S)	$5n$	$n - \log_2(\lvert G\rvert)$
Ours (Section 4.4)	DDH	bilinear (A)	$4n$	$n - \log_2(\lvert G\rvert)$
Ours (Section 4.5)	2-LIN	bilinear (S)	n	$n - 2\log_2(\lvert G\rvert)$
Ours (Section 4.4)	DDH	k-linear (A)	$2kn^{2/k}$	$n - \log_2(\lvert G\rvert)$
Ours (Section 4.5)	k-LIN	k-linear (S)	$n^{2/k}$	$n - k\log_2(\lvert G\rvert)$

Table 4.1: Efficiency characteristics of different LTDF constructions in the discrete log setting. As assumption in groups allowing (asymmetric) multilinear maps, DDH means the DDH assumption in every source group. "Type of map" denotes the necessary multilinear map, where "A" means asymmetric, and "S" symmetric. The "size of public description" is measured in group elements.

4.2 Lossy Trapdoor Functions from Encrypted Matrices

In 2008, Peikert and Waters ([99]) introduced the concept of lossy trapdoor functions and realized them from both the (decisional) Diffie-Hellman and from lattice-based assumptions. Both constructions use matrices with component wise encrypted entries. We will review their generalized concept in this section.

Let Setup, KeyGen, Enc and Dec denote algorithms of an additively homomorphic public key encryption scheme with message space $\mathbb{Z}_p$. Let $\cdot$ denote the homomorphic operation, i.e., $\mathsf{Enc}_{pk}(m;r) \cdot \mathsf{Enc}_{pk}(m';r') = \mathsf{Enc}_{pk}(m+m';r'')$, where the first parameter is the message and the second parameter is the encryption randomness. We require that r'' depends solely on r and r' and the encryption scheme remains secure when randomness is reused.[9] For convenience, we write $\mathsf{Enc}_{pk}(m;r)^x := \prod_{i=1}^{x} \mathsf{Enc}_{pk}(m;r)$. Additionally, observe that the encryption scheme allows another type of homomorphic operation, namely combining encryptions using the same randomness as follows (we will also write this operation multiplicatively):

$$\begin{aligned}\mathsf{Enc}_{pk}(m;r)\mathsf{Enc}_{pk'}(m';r) &= ([r], pk^r[m])([r], pk'^r[m']) \stackrel{\text{def}}{=} \\ ([r], pk^r[m]pk'^r[m']) &= \mathsf{Enc}_{pk\cdot pk'}(m+m';r).\end{aligned}$$

We also extend the definition of the "matrix-vector product in the exponent" $*$ from Section 2.1 to an "encrypted matrix-vector product" in the obvious way. That is, for $\vec{y} = (y_i) = (\mathsf{Enc}_{pk_i}(x_i;r_i))$ and a matrix $\mathbf{M} = (m_{i,j})$ over $\mathbb{Z}_p$, define $\vec{y} * \mathbf{M} := (\prod_i \mathsf{Enc}_{pk_i}(x_i;r_i)^{m_{i,j}})_j$; analogously for $\mathbf{M} * \vec{y}$, and $\mathbf{C} * \vec{x}$ and $\vec{x} * \mathbf{C}$ for an encrypted matrix $\mathbf{C}$ and a vector $\vec{x}$ over $\mathbb{Z}_p$. Note that $*$ is computed using the homomorphic operation. We will solely deal with terms where only one of the two possible types of homomorphic operations occur.

Let $\mathcal{PP}$ denote the public parameters of the encryption scheme (generated by Setup). For input size $n = n(\lambda) \in \mathbb{N}$ we define a matrix of n^2 encryptions as follows:

$$\mathbf{C}_{\mathbf{M}}(\mathcal{PP}, \vec{pk}, \vec{r}) := \left(\mathsf{Enc}_{pk_j}(m_{i,j};r_i)\right)_{i,j\in\{1,\ldots,n\}},$$

where $\mathbf{M} \in \{\mathbf{0}_n, \mathbf{I}_n\}$, $(pk_i, sk_i)_{i\in\{1,\ldots,n\}} \leftarrow \mathsf{KeyGen}(\mathcal{PP})$, $(r_i)_{i\in\{1,\ldots,n\}}$ randomly chosen. For simplicity, if it follows from the context which keys and randomness are

[9] Like [69], who also explain (other) LTDF constructions from homomorphic encryption, we do not assume any particular homomorphic property concerning the randomness.

used when building the matrix, we will omit the parameters and write $\mathbf{C_M} := \mathbf{C_M}(\mathcal{PP}, \vec{pk}, \vec{r})$. We call a matrix of this form *PW-matrix* (where PW stands for Peikert and Waters).

By construction, $\mathbf{C_M}$ is an entry-wise encryption of either the zero matrix or the identity matrix, where encryptions in each row use the same randomness and in each column the entries are encrypted under the same public key as depicted below.

$$\mathbf{C_M} = \begin{array}{c} \begin{array}{ccc} pk_1 \downarrow & pk_2 \downarrow \ldots & \end{array} \\ \begin{pmatrix} \mathsf{Enc}_{pk_1}(m_{1,1}; r_1) & \mathsf{Enc}_{pk_2}(m_{1,2}; r_1) & \ldots \\ \mathsf{Enc}_{pk_1}(m_{2,1}; r_2) & \mathsf{Enc}_{pk_2}(m_{2,2}; r_2) & \ldots \\ \vdots & \vdots & \ddots \end{pmatrix} \end{array} \begin{array}{l} \leftarrow r_1 \\ \leftarrow r_2 \\ \vdots \end{array}$$

We define the four algorithms, $S_{inj}, S_{lossy}, F_{ltdf}$ and F_{ltdf}^{-1}:

- **Injective sampling.** S_{inj} takes as input $n \in \mathbb{N}$, runs $\mathsf{Setup}(\lambda)$ to obtain system parameters $\mathcal{PP}$, obtains $(pk_i, sk_i)_{i\in\{1,\ldots,n\}} \leftarrow \mathsf{KeyGen}(\mathcal{PP})$, draws random values $(r_i)_{i\in\{1,\ldots,n\}}$ for the encryption scheme, computes $\mathbf{C_I}$ and outputs the function description $\mathbf{C_I}$ and the trapdoor $td = (sk_i)_{i\in\{1,\ldots,n\}}$. (Recall the definition of the matrices $\mathbf{I}$ and $\mathbf{0}$ from Section 2.1.)

- **Lossy sampling.** S_{loss} takes as input $n \in \mathbb{N}$, runs $\mathsf{Setup}(\lambda)$ to obtain system parameters $\mathcal{PP}$, obtains $(pk_i, sk_i)_{i\in\{1,\ldots,n\}} \leftarrow \mathsf{KeyGen}(\mathcal{PP})$, draws random values $(r_i)_{i\in\{1,\ldots,n\}}$ for the encryption scheme and outputs the function description $\mathbf{C_0}$.

- **Function evaluation.** F_{ltdf} takes as input a description $\mathbf{C_M}$ and $x \in \{0,1\}^n$ and outputs $y := \vec{x} * \mathbf{C_M}$, where $\vec{x}$ is the input x written as row vector and $*$ denotes the special matrix-vector product as defined above. Hence, the result is a vector y with components

$$y_j := \prod_{i=1}^{n} \mathsf{Enc}_{pk_j}(m_{i,j}; r_i)^{x_i}.$$

- **Inversion.** F_{ltdf}^{-1} takes as inputs a tuple $(\mathbf{C_I}, \vec{sk})$ generated by S_{inj} and an image y. It then reconstructs x component wise by computing $x_j = \mathsf{Dec}_{sk_j}(y_j)$ for all $j \in \{1, \ldots, n\}$ and outputs x.

We now explain why the function evaluation algorithm F_{ltdf} is lossy when running with inputs generated by the lossy sampling algorithm S_{loss}. By definition, F_{ltdf} outputs a vector $\vec{y} = (y_j)_{j\in\{1,\ldots,n\}}$ with $y_j := \prod_{i=1}^{n} \mathsf{Enc}_{pk_j}(m_{i,j}; r_i)^{x_i}$, where the encryption randomness does not depend on j. Thus all encryptions in $\vec{y}$ use the same randomness, denoted by $r(\vec{r}, x)$ to point out the dependencies. In the lossy case $\vec{y}$ contains only encryptions of zeros, thus messages and public keys are independent of the input x. Therefore, there is only one value in $\vec{y}$ containing information about x, namely $r(\vec{r}, x)$. This means that only $|r(\vec{r}, x)|$ bits of information about x are revealed. If we require Setup to generate parameters $\mathcal{PP}$ where the encryption randomness comes from a group of size p, there are only p possible values y for 2^n possible inputs x. Therefore F_{ltdf}, when running with input $\mathbf{C_0}$ from S_{loss} and the parameters are chosen such that $\log_2(p) < n$, loses $n - \log_2(p)$ bits of information.

In the injective case, i.e., F_{ltdf} running with input $\mathbf{C_I}$ from S_{inj}, function evaluation is invertible because the vector $\vec{y}$ contains encryptions of all bits of x under the n different public keys, all using again the same randomness $r(\vec{r}, x)$. Knowledge of the secret keys therefore suffices to recover the single bits of x.

In the original paper ([99]), the authors instantiated their construction with an ElGamal-like encryption scheme based on DDH. The public description of the function is the matrix $\mathbf{C_M}$, resulting in rather large public parameters of size $\mathbf{O}(n^2)$. There have been efforts to compress these public parameters, namely by Boyen and Waters ([23]). Their idea is to move the encryptions to the target group of a pairing. They publish $\mathbf{O}(n)$ elements of the domain groups, which then can be combined using the pairing to compute the encrypted matrix. The main difficulty is to assure that the main diagonal remains uncomputable to avoid distinguishability of the two function types. The drawback of this method is that there is a natural linear lower bound for the size of the public parameters, because the main diagonal already has n elements and is contained in the public key.

4.3 New Methods for Compressing LTDFs from Encrypted Matrices

To further improve the size of the public parameters of a lossy trapdoor function from encrypted matrices we have to find a method to compress the main diagonal of the matrix $\mathbf{C_M}$. Our approach is to build $\mathbf{C_M}$ from smaller matrices $\mathbf{C'_M}$, $\mathbf{C''_M}$ using a tensor product. This way, a matrix of size $n \times n$ can be computed using two matrices of size $\sqrt{n} \times \sqrt{n}$, reducing the size of the lossy trapdoor function's public description to $\mathbf{O}(\sqrt{n}^2)$. As we will show, one can even apply this method several times to further compress the matrices. Note that the tensor product preserves the structure of public keys and randomness of $\mathbf{C'_M}$ and $\mathbf{C''_M}$, in the sense that the resulting matrix has products of encryptions using the same randomness in each row, under the same public keys in each column and thus is again a PW-matrix. Concretely,

$$\mathbf{C_M} := \mathbf{C'_M} \otimes_\phi \mathbf{C''_M} = \left(\mathsf{Enc}_{pk'_j}(m_{i,j}; r'_i)\right)_{i,j\in\{1,\ldots,n\}} \otimes_\phi \left(\mathsf{Enc}_{pk''_j}(m_{i,j}; r''_i)\right)_{i,j\in\{1,\ldots,n\}} =$$

$$\begin{array}{c} \begin{array}{cccc} pk'_1, pk''_1 \downarrow & & pk'_1, pk''_2 \downarrow \ldots & \\ \end{array} \\ \begin{pmatrix} \phi(\mathsf{Enc}_{pk'_1}(m_{1,1}; r'_1), \mathsf{Enc}_{pk''_1}(m_{1,1}; r''_1)) & \phi(\mathsf{Enc}_{pk'_1}(m_{1,1}; r'_1), \mathsf{Enc}_{pk''_2}(m_{1,2}; r''_1)) & \ldots & \leftarrow r'_1, r''_1 \\ \phi(\mathsf{Enc}_{pk'_1}(m_{1,1}; r'_1), \mathsf{Enc}_{pk''_1}(m_{2,1}; r''_2)) & \phi(\mathsf{Enc}_{pk'_1}(m_{1,1}; r'_1), \mathsf{Enc}_{pk''_2}(m_{2,2}; r''_2)) & \ldots & \leftarrow r'_1, r''_2 \\ \vdots & \vdots & \ddots & \vdots \end{pmatrix} \end{array}$$

for a suitable function ϕ and the straightforward extension of the generalized Kronecker product from Definition 2.2.1 to encryptions from a homomorphic encryption scheme.

There are two main problems that arise with this construction. First, in the lossy case, not all encryption schemes have the property that there is a function ϕ such that $\phi(\mathsf{Enc}_{pk}(0; r), \mathsf{Enc}_{pk'}(0; r'))$ is again an encryption of zero (so that $\mathbf{C'_0} \otimes_\phi \mathbf{C''_0}$ is still an encryption of the all-zero matrix). Second, observe that in the injective case $\mathbf{C'_I} \otimes_\phi \mathbf{C''_I}$ contains products of encryptions of 0 and 1. Depending on the encryption scheme and ϕ, those products of encryptions might not encrypt 0. Instead, they

encrypt some "cross terms" that might also depend on the randomness and public keys. Thus $\mathbf{C_I}$ might not preserve the plaintext matrix $\mathbf{I}$ properly and we will get non-zero encryptions of messages outside the main diagonal. The final plaintext matrix could even be of low rank, potentially making the function lossy. This inhibits the straightforward recovery of inputs using the techniques from Section 4.2.

Our first construction uses ElGamal encryption "in the exponent" to compute the encryptions in $\mathbf{C'_M}$ and $\mathbf{C''_M}$. Then $\mathbf{C_M}$ is obtained by choosing the function ϕ to be a pairing e and compute $\mathbf{C_M}$ as $\mathbf{C'_M} \otimes_e \mathbf{C''_M}$. This way, $\mathbf{C_M}$ is still a PW-matrix and encrypts the all-zero matrix in the lossy case. It remains to show that in the injective case we can still derive functions F_{ltdf}, F^{-1}_{ltdf} from $\mathbf{C_M}$. The overall idea is to keep track of and make up for all "message-randomness cross terms" that arise in the plaintext matrix of $\mathbf{C_M}$ when applying the tensor product. This construction is described in Section 4.4. Using the same techniques and a multilinear map, we can further compress the public description of our LTDF (see Section 4.4.2 for details).

In Section 4.5 we apply our technique of compressing matrices using a Kronecker product to a k-LIN-based lossy trapdoor function from [46].

4.4 Lossy Trapdoor Functions from DDH using Asymmetric Multilinear Maps

In this section we describe our lossy trapdoor function based on DDH and asymmetric multilinear maps. For simplicity, we start with explaining it using a bilinear map. We instantiate the generic construction from Section 4.2 with the DDH-based ElGamal-like encryption scheme from Definition 2.6.1, also used in [23]. We give now a formal description of our lossy trapdoor function. See Figure 4.1 for an informal overview over the four algorithms that it consists of.

INJECTIVE SAMPLING. To sample an injective function with input size n, the injective sampling algorithm runs a prime order bilinear group generator $\mathcal{G}_2(n - l)$ to obtain two groups G_1, G_2 of order p equipped with an asymmetric pairing. It then computes and outputs two PW-matrices $\mathbf{C'_M}((p, G_1, g_1), [\vec{z}']_1, \vec{r}') \in (G_1 \times G_1)^{\sqrt{n}\times\sqrt{n}}$, $\mathbf{C''_M}((p, G_2, g_2), g_2^{\vec{z}''}, \vec{r}'') \in (G_2 \times G_2)^{\sqrt{n}\times\sqrt{n}}$ of size $\sqrt{n}\times\sqrt{n}$, where $\mathbf{M}$ is a plaintext matrix that is arbitrary but invertible (e.g. $\mathbf{M} = \mathbf{I}$). The matrices are computed using the encryption scheme from Definition 2.6.1, i.e., using randomness $\vec{r}', \vec{r}'' \xleftarrow{R} \mathbb{Z}_p^{\sqrt{n}}$ and secret keys $\vec{z}', \vec{z}''$ obtained from $\mathsf{KeyGen}(p, G_1, g_1)$, respectively $\mathsf{KeyGen}(p, G_2, g_2)$, which means that $\vec{z}'$ and $\vec{z}''$ are also random vectors from $\mathbb{Z}_p^{\sqrt{n}}$.

$$\mathbf{C'_M} = \begin{pmatrix} ([r'_1]_1, [r'_1 z'_1 + m_{1,1}]_1) & ([r'_1]_1, [r'_1 z'_2 + m_{1,2}]_1) & \dots \\ ([r'_2]_1, [r'_2 z'_1 + m_{2,1}]_1) & ([r'_2]_1, [r'_2 z'_2 + m_{2,2}]_1) & \dots \\ \vdots & \vdots & \ddots \end{pmatrix}$$

$$\mathbf{C''_M} = \begin{pmatrix} ([r''_1]_2, [r''_1 z''_1 + m_{1,1}]_2) & ([r''_1]_2, [r''_1 z''_2 + m_{1,2}]_2) & \dots \\ ([r''_2]_2, [r''_2 z''_1 + m_{2,1}]_2) & ([r''_2]_2, [r''_2 z''_2 + m_{2,2}]_2) & \dots \\ \vdots & \vdots & \ddots \end{pmatrix}$$

Additionally, S_{inj} outputs the secret keys $\vec{z}', \vec{z}''$ as trapdoors.

- $S_{inj}(n,l) \longrightarrow (\mathbf{C'_M}, \mathbf{C''_M}, \vec{z}', \vec{z}'')$, where
  - $\mathbf{C'_M} := \mathbf{C'_M}((p, G_1, g_1), \vec{r}', \vec{z}') \in (G_1 \times G_1)^{\sqrt{n}\times\sqrt{n}}$, i.e., an entry $c'_{i,j}$ of $\mathbf{C'_M}$ is computed as $c'_{i,j} := \mathsf{Enc}_{[\vec{z}'_j]_1}(m_{i,j}; \vec{r}'_i) = ([r'_i]_1, [r'_i z'_j + m_{i,j}]_1)$
  - $\mathbf{C''_M} := \mathbf{C''_M}((p, G_2, g_2), \vec{r}'', \vec{z}'') \in (G_2 \times G_2)^{\sqrt{n}\times\sqrt{n}}$
  - $\vec{r}', \vec{r}'', \vec{z}', \vec{z}'' \xleftarrow{\$} \mathbb{Z}_p^{1\times\sqrt{n}}$
  - $\mathbf{M} \leftarrow \mathbb{Z}_p^{\sqrt{n}\times\sqrt{n}}$ with $\mathrm{rank}(\mathbf{M}) = \sqrt{n}$, $\mathbf{M} =: (m_{i,j})_{i,j\in\{1,\ldots,\sqrt{n}\}}$
  - $(2, G_1, G_2, G_T, e, p, g_1, g_2) \longleftarrow \mathcal{G}_2(n-l)$
- $S_{loss}(n,l) \longrightarrow (\mathbf{C'_0}, \mathbf{C''_0})$, where
  - $\mathbf{C'_0} := \mathbf{C'_0}((p, G_1, g_1), \vec{z}', \vec{r}') \in (G_1 \times G_1)^{\sqrt{n}\times\sqrt{n}}$
  - $\mathbf{C''_0} := \mathbf{C''_0}((p, G_2, g_2), \vec{z}'', \vec{r}'') \in (G_2 \times G_2)^{\sqrt{n}\times\sqrt{n}}$
  - $\vec{r}', \vec{r}'', \vec{z}', \vec{z}'' \xleftarrow{\$} \mathbb{Z}_p^{1\times\sqrt{n}}$
  - $(2, G_1, G_2, G_T, e, p, g_1, g_2) \longleftarrow \mathcal{G}_2(n-l)$
- $F_{ltdf}(\mathbf{C'_M}, \mathbf{C''_M}, x) \longrightarrow y$, where
  - $y := x * \mathbf{C}_{\widetilde{\mathbf{M}}}$, i.e., $y_j = \prod_{i=1}^{n} c_{i,j}^{x_i}$ with $(c_{i,j})_{i,j\in\{1,\ldots,n\}} := \mathbf{C}_{\widetilde{\mathbf{M}}}$
  - $\mathbf{C}_{\widetilde{\mathbf{M}}} := \mathbf{C'_M} \otimes_e \mathbf{C''_M} \in G^{n\times n}$
- $F^{-1}_{ltdf}(\mathbf{C}_{\widetilde{\mathbf{M}}}, \vec{z}', \vec{z}'', y) \longrightarrow x$, where
  - $x := \mathsf{Dec}_{\vec{z}\widetilde{\mathbf{M}}^{-1}}(y * \widetilde{\mathbf{M}}^{-1})$, $\vec{z} := \vec{z}' \otimes \vec{z}'' \in \mathbb{Z}_p^{1\times n}$
  - $\widetilde{\mathbf{M}} := (\tilde{m}_{i,j})_{i,j\in\{1,\ldots,n\}}$ with $\tilde{m}_{i,j} := \mathsf{Dec}_{z_j}(c_{i,j})$
  - $(c_{i,j})_{i,j\in\{1,\ldots,n\}} := \mathbf{C}_{\widetilde{\mathbf{M}}}$

Figure 4.1: A formal description of our LTDF using ElGamal-like encryption based on DDH.

LOSSY SAMPLING. The lossy sampling algorithm does the same as the injective sampling algorithm, except it computes the PW-matrices as encryptions of the all-zero matrix (i.e., $\mathbf{M} = \mathbf{0}$) and does not output the secret keys.

FUNCTION EVALUATION. The evaluation algorithm first computes a PW-matrix $\mathbf{C}_{\widetilde{\mathbf{M}}}$ of size n by combining the smaller PW-matrices of the sampling algorithms using the tensor product on bilinear groups. The randomness and keys used in $\mathbf{C}_{\widetilde{\mathbf{M}}}$ can be computed from the randomness and keys of the smaller matrices. Then the algorithm proceeds as in the generic construction, i.e., computes and outputs $y := x * \mathbf{C}_{\widetilde{\mathbf{M}}}$ for a binary input vector x.

INVERSION. Observe that inverting an output y of F_{ltdf} is not as straightforward as in the generic construction, because S_{inj} did not encrypt the identity matrix but an invertible matrix $\mathbf{M}$. Even worse, this $\mathbf{M}$ is not preserved by the tensor product because of the aforementioned cross terms. Thus, $\mathbf{C}_{\widetilde{\mathbf{M}}}$ generally encrypts a matrix $\widetilde{\mathbf{M}} \neq \mathbf{M} \otimes \mathbf{M}$. However, note that the encryption scheme allows us to multiply an encrypted matrix $\widetilde{\mathbf{M}}$ with another matrix, e.g., with $\widetilde{\mathbf{M}}^{-1}$. The result is the encrypted identity matrix. Now $\widetilde{\mathbf{M}}$ can in fact be computed from $\mathbf{C}_{\widetilde{\mathbf{M}}}$ using the trapdoor (the secret keys used in the encryptions of $\mathbf{C}_{\widetilde{\mathbf{M}}}$). If $\widetilde{\mathbf{M}}$ is invertible (see remark below), $\widetilde{\mathbf{M}}^{-1}$ exists and we can compute $y' := y * \widetilde{\mathbf{M}}^{-1}$. Note that y' is equal to the image of x under a PW-matrix $\mathbf{C}_{\mathbf{I}_n}$ using different keys as in $\mathbf{C}_{\widetilde{\mathbf{M}}}$ (but the same randomness). Since those keys are computable, F^{-1}_{ltdf} can proceed with y' as in the generic construction.

4.4.1 Correctness and Security of our Construction

The construction requires the plaintext matrix $\widetilde{\mathbf{M}}$ of $\mathbf{C}_{\widetilde{\mathbf{M}}}$ to be invertible. Let $\mathbf{S}' := (r'_i z'_j)_{i,j\in\{1,\ldots,\sqrt{n}\}}$, $\mathbf{S}'' := (r''_i z''_j)_{i,j\in\{1,\ldots,\sqrt{n}\}}$ denote the two matrices containing the randomness and public keys used in $\mathbf{C}'_{\mathbf{M}}, \mathbf{C}''_{\mathbf{M}}$. Then it is easy to see that

$$\widetilde{\mathbf{M}} = \mathbf{M} \otimes \mathbf{M} + \mathbf{M} \otimes \mathbf{S}'' + \mathbf{S}' \otimes \mathbf{M},$$

where $\mathbf{M} \otimes \mathbf{S}'' + \mathbf{S}' \otimes \mathbf{M}$ are the cross terms that arise from the tensoring. Note that $\mathbf{S}', \mathbf{S}''$ are random matrices of rank 1 (up to a negligible statistical distance). Assuming the matrix $\widetilde{\mathbf{M}}$ to be invertible in this setting seems quite reasonable, and indeed we will see in Section 4.4.3 that this holds with overwhelming probability, even in a more general case.

Lemma 4.4.1. *For $\mathbf{M} \in \mathbb{Z}_p^{\sqrt{n}\times\sqrt{n}}$ with $\mathrm{rank}(\mathbf{M}) = \sqrt{n}$ and random matrices $\mathbf{S}', \mathbf{S}'' \in \mathbb{Z}_p^{\sqrt{n}\times\sqrt{n}}$ (each of rank 1), the matrix $\widetilde{\mathbf{M}} := \mathbf{M} \otimes \mathbf{M} + \mathbf{M} \otimes \mathbf{S}'' + \mathbf{S}' \otimes \mathbf{M}$ is invertible at least with probability $1 - (\frac{2}{p-1} + \frac{2}{p} + \frac{1}{p^{\sqrt{n}}})$.*

Proof. See Lemma 4.4.8 with $k = 2$. □

Lemma 4.4.2. *If the algorithm F_{ltdf} runs with matrices generated by $S_{loss}(n, l)$, it loses $l := n - \log_2(p)$ bits of information about the input vector $x \in \{0,1\}^n$.*

Proof. The proof is as in the generic construction. In the lossy case, y contains one group element that depends on x, namely $\mathbf{R}_x := g^{\sum_{i=1}^{n} r_i x_i}$ with $r_i := (c_{i,1})_1 \in G_T$ for $i \in \{1,\ldots,n\}$, $(c_{i,j})_{i,j\in\{1,\ldots,n\}} := \mathbf{C}_{\widetilde{\mathbf{M}}}$. Note that the element $\mathbf{R}_x$ is the first

element of each tuple in y because the same randomness is used in the rows of the PW-matrix $\mathbf{C}_{\widetilde{\mathbf{M}}}$. F_{ltdf}^{-1} therefore maps from $\{0,1\}^n$ to a subset of size $|G_T| = p$, and thus loses $n-(n-l) = l$ bits of information. □

Lemma 4.4.3. *The scheme is correct in the sense that if the algorithm F_{ltdf} runs with input vector x and matrices generated by S_{inj}, the inversion algorithm recovers x except with negligible probability over the choice of f.*

Proof. Remember that the encryption scheme is additively homomorphic and thus satisfies

$$\mathsf{Enc}_{[z]}(m;r)^{m'} = ([r]^{m'}, [rz+m]^{m'}) = ([rm'], [rzm'+mm']) = \mathsf{Enc}_{[z]}(mm'; rm').$$

Moreover, remember that the encryption scheme allows combining encryptions using the same randomness as follows (which we also write multiplicatively):

$$\begin{aligned}\mathsf{Enc}_{[z_1]}(m_1;r)\mathsf{Enc}_{[z_2]}(m_2;r) &= ([r],[z_1r+m_1])([r],[z_2r+m_2]) \\ &= ([r],[z_1r+m_1][z_2r+m_2]) \\ &= \mathsf{Enc}_{[z_1+z_2]}(m_1+m_2;r).\end{aligned}$$

We derive some basic rules from these properties. Let $\mathbf{Z} \in \mathbb{Z}_p^{n\times n}$ be a matrix where every row is equal to $\vec{z} \in \mathbb{Z}_p^{1\times n}$, $\mathbf{R}, \mathbf{M} \in \mathbb{Z}_p^{\sqrt{n}\times\sqrt{n}}$ arbitrary matrices and $\vec{x} \in \mathbb{Z}_p^{1\times n}$. Then it follows that

$$\vec{x} * \mathsf{Enc}_{[\mathbf{Z}]}(\mathbf{M};\mathbf{R}) = \mathsf{Enc}_{[\vec{z}]}(\vec{x}\mathbf{M}; \vec{x}\mathbf{R}). \tag{4.2}$$

Similarly, if $\vec{r} \in \mathbb{Z}_p^{1\times n}$ is a vector with equal entries, $\vec{z}, \vec{y} \in \mathbb{Z}_p^{1\times n}$ arbitrary vectors and $\mathbf{M} \in \mathbb{Z}_p^{n\times n}$ an arbitrary matrix, then

$$\mathsf{Enc}_{[\vec{z}]}(\vec{y};\vec{r}) * \mathbf{M} = \mathsf{Enc}_{[\vec{z}]*\mathbf{M}}(\vec{y}\mathbf{M};\vec{r}). \tag{4.3}$$

For correctness we have to show that for every output $(\mathbf{C}'_{\mathbf{M}}, \mathbf{C}''_{\mathbf{M}}, \vec{z}', \vec{z}'')$ of S_{inj} we have $F_{ltdf}^{-1}(\mathbf{C}_{\widetilde{\mathbf{M}}}, \vec{z}', \vec{z}'', F_{ltdf}(\mathbf{C}'_{\mathbf{M}}, \mathbf{C}''_{\mathbf{M}}, x)) = x$. Let $\mathbf{R} \in \mathbb{Z}_p^{n\times n}$ be the matrix containing the randomness used in the encryptions of $\mathbf{C}_{\widetilde{\mathbf{M}}}$ (note that $\mathbf{R}$ has equal columns) and similarly $\mathbf{Z}$ be the matrix containing all secret keys used in $\mathbf{C}_{\widetilde{\mathbf{M}}}$ (every row of $\mathbf{Z}$ is equal to $\vec{z}$ with $\vec{z} := \vec{z}' \otimes \vec{z}''$). Using the notation and rules from above we get

$$\begin{aligned}F_{ltdf}^{-1}(\mathbf{C}_{\widetilde{\mathbf{M}}}, \vec{z}', \vec{z}'', F_{ltdf}(\mathbf{C}'_{\mathbf{M}}, \mathbf{C}''_{\mathbf{M}}, x)) &= F_{ltdf}^{-1}(\mathbf{C}_{\widetilde{\mathbf{M}}}, \vec{z}', \vec{z}'', x * \mathbf{C}_{\widetilde{\mathbf{M}}}) \\ &= F_{ltdf}^{-1}(\mathbf{C}_{\widetilde{\mathbf{M}}}, \vec{z}', \vec{z}'', x * \mathsf{Enc}_{[\mathbf{Z}]}(\widetilde{\mathbf{M}};\mathbf{R})) \\ &\overset{(4.2)}{=} F_{ltdf}^{-1}(\mathbf{C}_{\widetilde{\mathbf{M}}}, \vec{z}', \vec{z}'', \mathsf{Enc}_{[\vec{z}]}(x\widetilde{\mathbf{M}}; x\mathbf{R})) \\ &= \mathsf{Dec}_{\vec{z}\widetilde{\mathbf{M}}^{-1}}(\mathsf{Enc}_{[\vec{z}]}(x\widetilde{\mathbf{M}}; x\mathbf{R}) * \widetilde{\mathbf{M}}^{-1}) \\ &\overset{(4.3)}{=} \mathsf{Dec}_{\vec{z}\widetilde{\mathbf{M}}^{-1}}(\mathsf{Enc}_{[\vec{z}]*\widetilde{\mathbf{M}}^{-1}}(x\widetilde{\mathbf{M}}\widetilde{\mathbf{M}}^{-1}; x\mathbf{R})) \\ &= \mathsf{Dec}_{\vec{z}\widetilde{\mathbf{M}}^{-1}}(\mathsf{Enc}_{[\vec{z}\widetilde{\mathbf{M}}^{-1}]}(x; x\mathbf{R})) \\ &= x.\end{aligned}$$

Note that, since $\mathbf{R}$ has equal columns, $x\mathbf{R}$ is a vector with equal entries. This is an essential condition for applying rule (4.3). Also note that F_{ltdf}^{-1} is only computable if $\widetilde{\mathbf{M}}$ is invertible. Applying Lemma 4.4.1, it follows that F_{ltdf}^{-1} outputs x with probability $1-(\frac{2}{p-1}+\frac{2}{p}+\frac{1}{p^{\sqrt{n}}})$. □

The next Lemma shows computational indistinguishability of the $\sqrt{n} \times \sqrt{n}$-sized matrices generated by S_{loss} and S_{inj} under the DDH assumption. We will need this result to show security of our construction, i.e., indistinguishability of the $n \times n$-sized matrices computed by F_{ltdf}.

Lemma 4.4.4. *If the groups generated by $\mathcal{G}_2$ satisfy the DDH assumption, the matrices $\boldsymbol{C}'_{\mathbf{0}}, \boldsymbol{C}''_{\mathbf{0}}$ generated by S_{loss} are computationally indistinguishable from the matrices $\boldsymbol{C}'_{\mathbf{M}}, \boldsymbol{C}''_{\mathbf{M}}$ generated by S_{inj}, for any (independent) distribution of the plaintext matrix $\mathbf{M}$.*

Proof. This result is not surprising since the case $\mathbf{M} = \mathbf{I}$ has already been proven in [23]. We split the proof in three parts. Let $(2, G_1, G_2, G_T, e, p, g_1, g_2)$ denote the output of $\mathcal{G}_2$. We denote the public parameters of the groups G_1 and G_2 by $\mathcal{PP}_i := (G_i, p, g_i)$ for $i \in \{1, 2\}$. First, we show that in each of the groups G_1, G_2, two PW-matrices encrypting either a uniformly random matrix $\mathbf{Q}$ or the all-zero matrix are computationally indistinguishable, i.e., for $i \in \{1, 2\}$ and $n \in \mathbb{N}$ it holds that

$$\mathbf{C_0}(\mathcal{PP}_i, \vec{pk}, \vec{r}) \overset{c}{\approx} \mathbf{C_Q}(\mathcal{PP}_i, \vec{pk}, \vec{r}), \tag{4.4}$$

where $\vec{pk} \in \mathbb{Z}_p^n$ is taken from $(pk_i, sk_i)_{i \in \{1,\ldots,n\}} \leftarrow \mathsf{KeyGen}(PP_i)$, $r \xleftarrow{R} \mathbb{Z}_p^n$ and $\mathbf{Q} \xleftarrow{R} \mathbb{Z}_p^{n \times n}$.

Next, we show that for $i \in \{1, 2\}$ and $n \in \mathbb{N}$ it holds that

$$(\mathbf{M}, \mathbf{C_0}(\mathcal{PP}_i, \vec{pk}, \vec{r})) \overset{c}{\approx} (\mathbf{M}, \mathbf{C_M}(\mathcal{PP}_i, \vec{pk}, \vec{r})), \tag{4.5}$$

where $\vec{pk}$ is again generated by KeyGen, $r \xleftarrow{R} \mathbb{Z}_p^n$ and $\mathbf{M}$ is generated according to the given plaintext distribution.

We conclude the proof of Lemma 4.4.4 by showing that for $n \in \mathbb{N}$

$$(\mathbf{C}'_{\mathbf{0}}(\mathcal{PP}_1, \vec{pk}', \vec{r}'), \mathbf{C}''_{\mathbf{0}}(\mathcal{PP}_2, \vec{pk}'', \vec{r}'')) \overset{c}{\approx} (\mathbf{C}'_{\mathbf{M}}(\mathcal{PP}_1, \vec{pk}', \vec{r}'), \mathbf{C}''_{\mathbf{M}}(\mathcal{PP}_2, \vec{pk}'', \vec{r}'')), \tag{4.6}$$

where $\vec{pk}', \vec{pk}''$ are again generated by KeyGen and $\vec{r}', \vec{r}'' \xleftarrow{R} \mathbb{Z}_p^n$.

PROOF OF EQUATION (4.4). For a proof of the statement in one of the groups, e.g. G_1, we define a series of $n + 1$ subsequent games Game_k, $k \in \{0, \ldots, n\}$. Game_k contains a tuple $(A, [\vec{z}]_1, \vec{r})$ with $\vec{z}, \vec{r} \xleftarrow{R} \mathbb{Z}_p^n$ and a matrix $A = (a_{i,j})_{i,j \in \{1,\ldots,n\}}$ with tuples $a_{i,j}$ as entries defined as follows:

$$a_{i,j} = ([r_i]_1, \mathbf{R}_{i,j}) \text{ with } \mathbf{R}_{i,j} \xleftarrow{R} G_1 \text{ for } i \leq k$$
$$a_{i,j} = ([r_i]_1, [r_i z_j]_1) \text{ for } i > k.$$

Observe that in game Game_n we have $A = \mathbf{C_Q}(\mathcal{PP}_1, [\vec{z}]_1, \vec{r})$ for a random $\mathbf{Q} \in G_1^{n \times n}$ and in game Game_0 we have $A = \mathbf{C_0}(\mathcal{PP}_1, [\vec{z}]_1, \vec{r})$.

Now let $([1]_1, [x]_1, [y]_1, [z]_1)$ denote a DDH challenge tuple. We show how we can use a distinguisher between two subsequent games to solve the DDH challenge. For this we first draw $\vec{a}, \vec{b} \xleftarrow{R} \mathbb{Z}_p^n$. We use $\vec{a}, \vec{b}$ to re-randomize the DDH challenge tuple as follows: for each $i \in \{1, \ldots, n\}$ we compute a new DDH tuple as $([1]_1, [x]_1, [ya_i + b_i]_1, [xb_i + za_i]_1)$. Note that if $z = xy$, the new tuples contain n encryptions of zero

under n different (and random) public keys $[ya_i + b_i]_1 =: [s_i]_1$, $i \in \{1, \ldots, n\}$. If $z \neq xy$, the new tuples contain n encryptions of random values $(z - xy)a_i$, $i \in \{1, \ldots, n\}$. We then choose $k \in \{1, \ldots, n\}$, draw random elements $\vec{r} \xleftarrow{R} \mathbb{Z}_p^n$ and $\mathbf{R}_{i,j} \xleftarrow{R} G_1$ and compute the matrix A as

$$A := \begin{pmatrix} ([r_1]_1, \mathbf{R}_{1,1}) & ([r_1]_1, \mathbf{R}_{1,2}) & \ldots & ([r_1]_1, \mathbf{R}_{1,n}) \\ \vdots & \vdots & \ldots & \vdots \\ ([r_{k-1}]_1, \mathbf{R}_{k-1,1}) & ([r_{k-1}]_1, \mathbf{R}_{k-1,2}) & \ldots & ([r_{k-1}]_1, \mathbf{R}_{k-1,n}) \\ ([x]_1, [xb_1 + za_1]_1) & ([x]_1, [xb_2 + za_2]_1) & \ldots & ([x]_1, [xb_n + za_n]_1) \\ ([r_{k+1}]_1, [r_{k+1}s_1]_1) & ([r_{k+1}]_1, [r_{k+1}s_2]_1) & \ldots & ([r_{k+1}]_1, [r_{k+1}s_n]_1) \\ \vdots & \vdots & \ldots & \vdots \end{pmatrix}.$$

Note that we implicitly set the randomness in row i to $r_i := x$ and the public key in each column $j \in \{1, \ldots, n\}$ to $[s_j]_1$. The matrix contains encryptions of random elements in row k if $z \neq xy$ and encryptions of zero otherwise. Thus, if $z \neq xy$ we are in game Game_k and if $z = xy$ we are in game Game_{k-1}.

Since Game_0 and Game_n cannot be distinguishable without any of two subsequent games in between being distinguishable, we have shown that $\mathbf{C_0}(\mathcal{PP}_1, [\vec{z}]_1, \vec{r}) \overset{c}{\approx} \mathbf{C_Q}(\mathcal{PP}_1, [\vec{z}]_1, \vec{r})$.

PROOF OF EQUATION (4.5). We show $(\mathbf{M}, \mathbf{C_0}) \overset{c}{\approx} (\mathbf{M}, \mathbf{C_Q}) \overset{c}{\approx} (\mathbf{M}, \mathbf{C_M})$ for a random matrix $\mathbf{Q} \in \mathbb{Z}_p^{l \times l}$. The first indistinguishability follows directly from Equation (4.4). For showing $(\mathbf{M}, \mathbf{C_Q}) \overset{c}{\approx} (\mathbf{M}, \mathbf{C_M})$, we again use Equation (4.4) to obtain a PW-matrix $\mathbf{C}$ which encrypts $\mathbf{0}_l$ or a random matrix $\mathbf{Q}'$, draw a matrix $\mathbf{M}$ according to the given plaintext distribution, and compute the tuple $(\mathbf{M}, \mathbf{C} \cdot [\mathbf{M}])$. If $\mathbf{C} = \mathbf{C_0}$ we computed $(\mathbf{M}, \mathbf{C_M})$, else $(\mathbf{M}, \mathbf{C_Q})$. Note that $\mathbf{Q}$ is again a random matrix since $\mathbf{Q} = \mathbf{M} + \mathbf{Q}'$.

PROOF OF EQUATION (4.6). We define the following set of three games.

$$\begin{aligned} \mathrm{Game}_0 &: (\mathbf{C'_0}(\mathcal{PP}_1, \vec{pk}', \vec{r}'), \mathbf{C''_0}(\mathcal{PP}_2, \vec{pk}'', \vec{r}'')) \\ \mathrm{Game}_1 &: (\mathbf{C'_0}(\mathcal{PP}_1, \vec{pk}', \vec{r}'), \mathbf{C''_M}(\mathcal{PP}_2, \vec{pk}'', \vec{r}'')) \\ \mathrm{Game}_2 &: (\mathbf{C'_M}(\mathcal{PP}_1, \vec{pk}', \vec{r}'), \mathbf{C''_M}(\mathcal{PP}_2, \vec{pk}'', \vec{r}'')) \end{aligned}$$

Indistinguishability can be shown via a straightforward reduction from Eq. 4.5. Note that Eq. 4.5 also contains the plaintext matrix $\mathbf{M}$, which makes it easy to compute $\mathbf{C''_M}$ on the right hand side of Game_1 and Game_2. □

Putting it all together we obtain the following result:

Theorem 4.4.5. *If the groups generated by $\mathcal{G}_2(n - l)$ satisfy the* DDH *assumption, the algorithms S_{inj}, S_{loss}, F_{ltdf} and F_{ltdf}^{-1} define a collection of (n,l)-lossy trapdoor functions.*

Proof. See Lemma 4.4.2, Lemma 4.4.3, Lemma 4.4.4. □

4.4.2 Efficiency and Optimizations

For evaluating inputs of size n, the public description of our LTDF from DDH using bilinear groups is given by two PW-matrices of size $\sqrt{n} \times \sqrt{n}$. Since we have two group elements per ciphertext, the matrices consist of $4\sqrt{n}^2 = 4n$ group elements. This is asymptotically as efficient as the constructions of Boyen and Waters ([23]), which also have a public description of linear size. But in contrast to the construction from [23], our LTDF's public description can be further compressed using multilinear maps of higher order than 2 in a straightforward way. Namely, we apply the tensor product several times and build our final PW-matrix recursively from smaller PW-matrices. Furthermore, since $\otimes$ is associative and hence $\otimes_e$ just as well, we can evaluate the tensor products in an arbitrary order. Using a k-linear map for some $k \in \mathbb{N}$ we can start combining 2 matrices $\mathbf{C}_{\mathbf{M}}^{(1)}$ and $\mathbf{C}_{\mathbf{M}}^{(2)}$ and successively append matrices $\mathbf{C}_{\mathbf{M}}^{(i)}$ for $i = 3, \ldots, k$ from the right, i.e., computing $(((\mathbf{C}_{\mathbf{M}}^{(1)} \otimes_e \mathbf{C}_{\mathbf{M}}^{(2)}) \otimes_e \mathbf{C}_{\mathbf{M}}^{(3)}) \otimes_e \ldots)$....

If we again denote the matrices containing secret keys and randomness used in $\mathbf{C}_{\mathbf{M}}^{(i)}$ with $\mathbf{S}_i$ for $i = 1, \ldots, k$ for some $k \in \mathbb{N}$, then the plaintext matrix $\mathbf{L}_i$ of the intermediate result after i matrices have been combined can be computed using the following recursive formula:

$$(\mathbf{L}_1, \mathbf{R}_1) := (\mathbf{M}, \mathbf{S}_1)$$
$$(\mathbf{L}_i, \mathbf{R}_i) := (\mathbf{L}_{i-1} \otimes \mathbf{M} + \mathbf{L}_{i-1} \otimes \mathbf{S}_i + \mathbf{R}_{i-1} \otimes \mathbf{M}, \mathbf{R}_{i-1} \otimes \mathbf{S}_i).$$

We will analyze the invertibility of the matrix $\mathbf{L}_k$, the plaintext matrix of the final PW-matrix, in the next section.

We now examine the size of the public description of our general construction. Using an asymmetric k-linear map we can combine k PW-matrices of size $\sqrt[k]{n} \times \sqrt[k]{n}$ each using different randomness and public keys and obtain a matrix of size $n \times n$. If the matrix $\mathbf{M}$ is invertible, with overwhelming probability the resulting matrix describes an injective function (see Lemma 4.4.8). If $\mathbf{M} = \mathbf{0}$, the resulting matrix describes a lossy function. The proofs of lossiness, invertibility and indistinguishability can be directly translated from the bilinear to the multilinear setting. Overall we obtain a lossy trapdoor function using a k-linear map with a public description size in $\mathbf{O}(k(\sqrt[k]{n})^2)$.

4.4.3 Construction of Large Invertible Matrices from few Randomness

In this chapter, we provide evidence for the invertibility of our lossy trapdoor function. Namely, we show that if we start with "small" invertible matrices and recursively combine them as described in Section 4.4.2, we obtain a PW-matrix encrypting a plaintext matrix that is invertible with overwhelming probability.

Lemma 4.4.6. *Let $\mathbf{Q} \in \mathbb{Z}_p^{n \times n}$ be of rank 1, let $\mathbf{M} \in \mathbb{Z}_p^{n \times n}$ be invertible and let $z \in \mathbb{Z}_p$. Then it holds:*

1. *If $\mathbf{Q}$ is uniformly random (but still a rank-1 matrix) and $\mathbf{M}$ is constant, then the matrix $\mathbf{M} + \mathbf{Q}$ is invertible with probability $1 - \frac{1}{p-1}$.*
2. *If $\mathbf{Q}$ and $\mathbf{M}$ are constant, and z is uniformly random, then the matrix $z \cdot \mathbf{M} + \mathbf{Q}$ is invertible at least with probability $1 - \frac{2}{p}$.*

Proof. W.l.o.g., $\mathbf{M}$ is the identity matrix—otherwise we perform an according basis transformation. Since $\mathrm{rank}(\mathbf{Q}) = 1$, we have that 0 is an eigenvalue of $\mathbf{Q}$ with multiplicity $n-1$ and thus the characteristic polynomial of $\mathbf{Q}$ can be written as $x^{n-1}(x-\lambda)$ with $\lambda \in \mathbb{Z}_p \backslash \{0\}$. In other words, every rank-1 matrix has exactly one non-zero eigenvalue. In the following, let λ always denote this eigenvalue of $\mathbf{Q}$. Further, note that the following four assertions are equivalent:

- The matrix $z \cdot \mathbf{I} + \mathbf{Q}$ is not invertible.
- There exists a vector $x \neq 0$, such that $(z \cdot \mathbf{I} + \mathbf{Q})x = \mathbf{0}$.
- There exists a vector $x \neq 0$, such that $\mathbf{Q}x = -zx$.
- We have that $-z$ is an eigenvalue of $\mathbf{Q}$.

With this in mind, we can easily prove the two claims our lemma consists of.

1. If $\mathbf{Q}$ is uniformly random, then by symmetry reasons its unique non-zero eigenvalue λ must also be uniformly random over $\mathbb{Z}_p \backslash \{0\}$. This yields:

$$\Pr[\mathbf{I} + \mathbf{Q} \text{ is not invertible}] \;=\; \Pr[-1 = \lambda] \;=\; \tfrac{1}{p-1}.$$

2. If $\mathbf{Q}$ is constant and z is uniformly random, we have:

$$\Pr[z \cdot \mathbf{I} + \mathbf{Q} \text{ is not invertible}] \;\leq\; \Pr[-z \in \{0, \lambda\}] \;=\; \tfrac{2}{p}. \qquad \square$$

Lemma 4.4.7. *Let $T \in \mathbb{Z}_p^{n \times n}$ be uniformly random of rank 1. Then, the statistical distance between $\mathrm{tr}(T)$ and uniform randomness is upper bounded by $\frac{1}{p^n}$.*

Proof. First of all, note that we can write T as the outer product of two vectors $r, s \in \mathbb{Z}_p^n$, which are uniformly random subject to the sole condition that none of them is all-zero. In particular, r and s are statistically independent of each other. Further, note that $\mathrm{tr}(T)$ is just the inner product of r and s, which we henceforth denote by $\langle r|s\rangle$. Moreover, let the statistical distance of any two random variables X, Y be denoted by $\Delta(X, Y)$.

Now, let r' be uniformly random over $\mathbb{Z}_p^n$ (including the all-zero vector). Hence, $\langle r'|s\rangle$ is uniformly distributed over $\mathbb{Z}_p$. I.e., the statistical distance of $\mathrm{tr}(T)$ from uniform randomness can be written as $\Delta(\langle r|s\rangle, \langle r'|s\rangle)$. However, by construction it holds:

$$\Delta(\langle r|s\rangle, \langle r'|s\rangle) \;\leq\; \Delta((r,s),(r',s)) \;=\; \Delta(r, r') \;=\; \tfrac{1}{p^n}. \qquad \square$$

Lemma 4.4.8. *Let $\mathbf{M}_1, \mathbf{S}_1, \ldots, \mathbf{M}_k, \mathbf{S}_k \in \mathbb{Z}_p^{n \times n}$ be given, where the $\mathbf{M}_i$ are arbitrary but fixed full-rank matrices and the $\mathbf{S}_i$ are uniformly random of rank 1. Let $(\mathbf{L}_1, \mathbf{R}_1) := (\mathbf{M}_1, \mathbf{S}_1)$ and let inductively $\mathbf{L}_i := \mathbf{L}_{i-1} \otimes \mathbf{M}_i + \mathbf{L}_{i-1} \otimes \mathbf{S}_i + \mathbf{R}_{i-1} \otimes \mathbf{M}_i$ and $\mathbf{R}_i := \mathbf{R}_{i-1} \otimes \mathbf{S}_i$, where $\otimes$ denotes the Kronecker product. Then, $\mathbf{L}_k$ is invertible at least with the following probability:*

$$1 - \left(\frac{k}{p-1} + \frac{2(k-1)}{p} + \frac{k-1}{p^n} \right).$$

Proof. Our proof is by induction on k and consists of three steps. First, we compute the probability ρ_1 that $\mathbf{L}_1$ and $\mathbf{L}_1 + \mathbf{R}_1$ are invertible, which is the base case. Next, we compute the probability ρ_2 that $\mathbf{L}_i + \mathbf{R}_i$ is invertible, conditioned to the event that $\mathbf{L}_{i-1} + \mathbf{R}_{i-1}$ is already invertible. Finally, we estimate the probability ρ_3 that $\mathbf{L}_i$ is invertible, conditioned to the event that $\mathbf{L}_{i-1}$ and $\mathbf{L}_{i-1} + \mathbf{R}_{i-1}$ are both invertible. By the Union bound, the matrices $\mathbf{L}_1, \ldots, \mathbf{L}_k$ and $\mathbf{L}_1 + \mathbf{R}_1, \ldots, \mathbf{L}_k + \mathbf{R}_k$ are all simultaneously invertible at least with probability $1 - ((1-\rho_1) + (k-1)(1-\rho_2) + (k-1)(1-\rho_3))$, which will directly yield the assertion of our lemma.

Step 1 (computation of ρ_1). We show: The matrices $\mathbf{L}_1$ and $\mathbf{L}_1 + R_1$ are invertible with probability $1 - \frac{1}{p-1}$.

Since $\mathbf{L}_1 = \mathbf{M}_1$, this matrix is always invertible by assumption. Moreover, as $\mathbf{L}_1 + \mathbf{R}_1 = \mathbf{M}_1 + \mathbf{S}_1$, this matrix is invertible by Lemma 4.4.6.1 with probability $1 - \frac{1}{p-1}$, as claimed.

Step 2 (computation of ρ_2). We show: Given that $\mathbf{L}_{i-1} + \mathbf{R}_{i-1}$ is invertible, we have that $\mathbf{L}_i + \mathbf{R}_i$ is invertible with probability $1 - \frac{1}{p-1}$.

Note that by construction we have:

$$\begin{aligned} \mathbf{L}_i + \mathbf{R}_i &= \mathbf{L}_{i-1} \otimes \mathbf{M}_i + \mathbf{L}_{i-1} \otimes \mathbf{S}_i + \mathbf{R}_{i-1} \otimes \mathbf{M}_i + \mathbf{R}_{i-1} \otimes \mathbf{S}_i \\ &= (\mathbf{L}_{i-1} + \mathbf{R}_{i-1}) \otimes (\mathbf{M}_i + \mathbf{S}_i). \end{aligned}$$

Since $\mathbf{L}_{i-1} + \mathbf{R}_{i-1}$ is invertible by our induction hypothesis, we only have to show that $\mathbf{M}_i + \mathbf{S}_i$ is invertible. However, by Lemma 4.4.6.1 this is the case with probability $1 - \frac{1}{p-1}$, as claimed.

Step 3 (estimation of ρ_3). We show: Given that $\mathbf{L}_{i-1}$ and $\mathbf{L}_{i-1} + \mathbf{R}_{i-1}$ are both invertible, we have that $\mathbf{L}_i$ is invertible at least with probability $1 - \frac{2}{p} - \frac{1}{p^n}$.

As $\mathbf{M}_i$ is invertible by assumption and $\mathbf{L}_{i-1} + \mathbf{R}_{i-1}$ is invertible by our induction hypothesis, we can set:

$$T := \mathbf{S}_i \cdot \mathbf{M}_i^{-1} \qquad K := \mathbf{L}_{i-1} \cdot (\mathbf{L}_{i-1} + \mathbf{R}_{i-1})^{-1} \qquad \mathbf{L}' := \mathbf{I}_{n^i} + K \otimes T.$$

Hence, the matrix $\mathbf{L}_i = \mathbf{L}' \cdot ((\mathbf{L}_{i-1} + \mathbf{R}_{i-1}) \otimes \mathbf{M}_i)$ is invertible if and only if $\mathbf{L}'$ is invertible. In other words, we have to show that $\det(\mathbf{L}') \neq 0$. Note that we can write $T = U \cdot V$ with $U \in \mathbb{Z}_p^{n \times 1}$ and $V \in \mathbb{Z}_p^{1 \times n}$, since $\mathrm{rank}(T) = \mathrm{rank}(\mathbf{S}_i) = 1$. Also note that $\mathrm{tr}(T) = V \cdot U$. Let $t := \mathrm{tr}(T)$. It follows by Sylvester's determinant theorem:

$$\begin{aligned} \det(\mathbf{L}') = \det(\mathbf{I}_{n^i} + (K \otimes U) \cdot (\mathbf{I}_{n^{i-1}} \otimes V)) &= \det(\mathbf{I}_{n^{i-1}} + (\mathbf{I}_{n^{i-1}} \otimes V) \cdot (K \otimes U)) \\ &= \det(\mathbf{I}_{n^{i-1}} + t \cdot K). \end{aligned}$$

Thus, it suffices to show that $\mathbf{I}_{n^{i-1}} + t \cdot K$ is invertible, which by definition of K is equivalent to $(t+1) \cdot \mathbf{L}_{i-1} + \mathbf{R}_{i-1}$ being invertible. However, t is statistically independent of $(\mathbf{L}_{i-1}, \mathbf{R}_{i-1})$ and $\frac{1}{p^n}$-close to uniform randomness by Lemma 4.4.7. Moreover, $\mathbf{L}_{i-1}$ is invertible by our induction hypothesis and $\mathrm{rank}(\mathbf{R}_{i-1}) = \mathrm{rank}(\mathbf{S}_{i-1} \otimes \ldots \otimes \mathbf{S}_1) = \mathrm{rank}(\mathbf{S}_{i-1}) \cdot \ldots \cdot \mathrm{rank}(\mathbf{S}_1) = 1$. Combining all this with Lemma 4.4.6.2, we can finally conclude that $(t+1) \cdot \mathbf{L}_{i-1} + \mathbf{R}_{i-1}$ is invertible at least with probability $1 - \frac{2}{p} - \frac{1}{p^n}$. □

- $S_{inj}(\lambda, n) \longrightarrow ([\mathbf{M}], \mathbf{M})$, where
  - $\mathbf{M} \xleftarrow{R} \mathbf{R}_n(\mathbb{Z}_p^{n\times n})$
  - $(g, G, p) \longleftarrow \mathcal{G}_1(\lambda)$
- $S_{loss}(\lambda, n, l) \longrightarrow [\mathbf{M}]$, where
  - $\mathbf{M} \xleftarrow{R} \mathbf{R}_k(\mathbb{Z}_p^{n\times n})$
  - $(g, G, p) \longleftarrow \mathcal{G}_1(\lambda)$
- $F_{ltdf}([\mathbf{M}], x) \longrightarrow y$ with $y := [\mathbf{M}] * x$
- $F_{ltdf}^{-1}(\mathbf{M}, y) \longrightarrow x$ with $x := \mathrm{DLOG}_g(\mathbf{M}^{-1} * y) \in \{0,1\}^n$, where the discrete logarithm $\mathrm{DLOG}_g(h)$ of a group element is computed by brute force[a]

[a] Note that since this discrete logarithm will always only be a bit, this can be done efficiently.

Figure 4.2: A formal description of the LTDF based on d-LIN from [46].

4.5 Lossy Trapdoor Functions from k-LIN using Symmetric Multilinear Maps

Freeman et al. ([46]) constructed an LTDF based on k-LIN with a public description that is also a matrix. Although they do not use the same technique as Peikert and Waters with structured keys and randomness, their approaches have certain similarities. In this chapter we will describe their construction and give a method to compress the public parameters of their lossy trapdoor function using symmetric multilinear maps.

4.5.1 Lossy Trapdoor Functions based on k-LIN

We informally describe the construction of an LTDF based on k-LIN from [46]. Similar to Peikert and Waters, they use the fact that a map $x \mapsto \mathbf{C}x$ with a matrix $\mathbf{C}$ is injective if $\mathbf{C}$ is invertible. If $\mathbf{C}$ is singular, the lossiness depends on the rank of $\mathbf{C}$ and the number of group elements. When $\mathbf{C}$ is simply a matrix $[\mathbf{M}]$ with $\mathbf{M}$ secret, then the d-LIN assumption implies indistinguishability of $[\mathbf{M}]$ and $[\mathbf{M}']$ where $\mathbf{M}$ has full rank and $\mathbf{M}'$ has rank d (see [95], Lemma A.1.).

For a more formal description let $\mathbf{R}_d(\mathbb{Z}_p^{n\times n})$ denote the set of $n\times n$-matrices defined over $\mathbb{Z}_p$ of rank d with $d \in \{1, \ldots, n\}$ and $n \in \mathbb{N}$. Let $\mathcal{G}$ be a group generator that, on input $\lambda \in \mathbb{N}$ outputs a tuple (p, G, g) where G is a group of order p with $|p| = \lambda$ and generator g. The LTDF of [46] is then formally defined in Figure 4.2.

Theorem 4.5.1 (Th. 6.2 from [46])**.** *If the k-LIN assumption holds in G, then the above algorithms define a collection of (n, l)-lossy trapdoor functions for $l := n - \log_2(p) \cdot k$.*

Observe that lossiness is guaranteed by the choice of the rank k of the lossy matrix. Namely, for a lossy matrix $\mathbf{M}$ of rank k, the vector $[\mathbf{M}] * x$ is contained in a subgroup of size p^k. For an injective matrix $\mathbf{M}$ there are 2^n possible vectors $[\mathbf{M}] * x$, since

- $S_{inj}(\lambda, n, l) \longrightarrow ([\mathbf{M}], \mathbf{M})$, where
  - $\mathbf{M} \xleftarrow{R} \mathbf{R}_{\sqrt[k]{n}}(\mathbb{Z}_p^{\sqrt[k]{n} \times \sqrt[k]{n}})$
  - $(k, G, G_T, e, p, g) \longleftarrow \mathcal{G}_k(\lambda)$
  - $k \in \mathbb{N}$ with $k = 2^i$ for some $i \in \mathbb{N}$ such that $\lambda = \lceil \frac{n-l}{k^k} \rceil$ (we have $k < \sqrt[k]{n}$ if $\lambda > 1$)
- $S_{loss}(\lambda, n, l) \longrightarrow ([\mathbf{M}])$, where
  - $\mathbf{M} \xleftarrow{R} \mathbf{R}_k(\mathbb{Z}_p^{\sqrt[k]{n} \times \sqrt[k]{n}})$
  - $(k, G, G_T, e, p, g) \longleftarrow \mathcal{G}_k(\lambda)$
  - $k \in \mathbb{N}$ with $k = 2^i$ for some $i \in \mathbb{N}$ such that $\lambda = \lceil \frac{n-l}{k^k} \rceil$ (we have $k < \sqrt[k]{n}$ if $\lambda > 1$)
- $F_{ltdf}([\mathbf{M}], x) \longrightarrow y$, where
  - $y := [\widetilde{\mathbf{M}}] * x$
  - $[\widetilde{\mathbf{M}}] := [\mathbf{M}] \otimes_e \cdots \otimes_e [\mathbf{M}] = [\mathbf{M} \otimes \cdots \otimes \mathbf{M}] \in G_T^{n \times n}$ (where the tensor product is applied $k-1$ times)
- $F_{ltdf}^{-1}(\mathbf{M}, y) \longrightarrow x$, where
  - $x := \widetilde{\mathbf{M}}^{-1} * y$
  - $\widetilde{\mathbf{M}} := \mathbf{M} \otimes \cdots \otimes \mathbf{M}$ (where the tensor product is applied $k-1$ times)

Figure 4.3: A formal description of the compressed LTDF based on the construction from [46].

injectivity assures exactly one image per possible input. Since $p^k = 2^{n-l}$, this leads to a lossiness of l bits as claimed.

Obviously the LTDF described in Figure 4.2 has a public description size in $\mathbf{O}(n^2)$, where n denotes the size of the binary input vector x.

4.5.2 Compressing the Public Description

We now compress the outputs of S_{inj} and S_{loss} using a tensor product in the exponents. We formally describe our modified version of the construction from [46]. Let $\mathcal{G}_*$ be the symmetric group generator from Definition 2.3.2. Our compressed LTDF is then described in Figure 4.3.

Recall that for the tensor product it holds that $\mathrm{rank}(\mathbf{M} \otimes \mathbf{M}') = \mathrm{rank}(\mathbf{M}) \cdot \mathrm{rank}(\mathbf{M}')$ and thus $\mathrm{rank}(\mathbf{M}) = \prod_{i=1}^{k} \mathrm{rank}(\mathbf{M}_i)$. Therefore, the rank of the lossy matrix $[\mathbf{M}]$ is k^k, and the lossiness of the final construction is $n - \log_2(p) \cdot k^k$. Indistinguishability of injective and lossy public keys follows directly from the k-LIN assumption in G. Correctness of invertibility can be shown as in the original construction.

For evaluating inputs of size n, our lossy trapdoor function described above thus has a public description size of $(\sqrt[k]{n})^2$ G-elements.

Chapter 5

New Composite-To-Prime-Order Transformations

In this chapter, we consider efficiency of cryptographic constructions using multilinear maps. We focus on a certain class of cryptosystems that make use of cyclic groups of composite order. In particular in combination with a multilinear map e, groups of composite order exhibit several interesting properties. (For instance, $e(\mathcal{P}_1, \mathcal{P}_2) = 1$ for elements $\mathcal{P}_1, \mathcal{P}_2$ of co-prime order. Or, somewhat more generally, the map operates on the different prime-order components of G independently.) Already in the case of pairings, this enables interesting technical applications (e.g., [114, 89]), but also comes at a price. Namely, to accommodate suitably hard computational problems, composite-order groups have to be chosen substantially larger than prime-order groups. Specifically, it should be hard to factor the group order. This leads to significantly slower operations in composite-order groups: [45] suggests that for realistic parameters, Tate pairings in composite-order groups are by a factor of about 50 less efficient than in prime-order groups. It is thus interesting to try to find substitutes for the technical features offered by composite-order groups in prime-order settings.

5.1 Overview

FREEMAN'S COMPOSITE-ORDER-TO-PRIME-ORDER TRANSFORMATION. Freeman [45] has offered a framework and tools to semi-generically convert cryptographic constructions from a composite-order to a prime-order setting. Similar transformations have also been implicit in previous works [63, 114]. The premise of Freeman's approach is that composite-order group elements "behave as" vectors over a prime field. In this interpretation, composite-order subgroups correspond to linear subspaces.

Moreover, we can think of the vector components as exponents of prime-order group elements; we can then associate, e.g., a composite-order subgroup indistinguishability problem with the problem of distinguishing vectors (chosen either from a subspace or the whole space) "in the exponent." More specifically, Freeman showed that the composite-order subgroup indistinguishability assumption can be implemented in a prime-order group with the Decisional Diffie-Hellman (or with the k-linear) assumption. A pairing operation over the composite-order group then translates into a suitable "multiplication of vectors," which can mean different things, depending on the desired properties. For instance, Freeman considers both an inner product and a Kronecker product as "vector multiplication" operations (of course with different effects).

LIMITATIONS OF FREEMAN'S APPROACH. Freeman's work has spawned a number of

follow-up results that investigate more general or more efficient conversions of this type [89, 108, 107, 85, 86]. We note that all of these works follow Freeman's interpretation of vectors, and even his possible interpretations of a vector multiplication. Unfortunately, during these investigations, certain lower bounds for the efficiency of these transformations became apparent. For example, Seo [107] proves lower bounds both for the computational cost and the dimension of the resulting vector space of *arbitrary* transformations in Freeman's framework. More specifically, Seo reports a concrete bound on the number of required prime-order pairing operations necessary to simulate a composite-order pairing.

However, of course, these lower bounds crucially use the vector-space interpretation of Freeman's framework. Specifically, it is conceivable that a (perhaps completely different) more efficient composite-order-to-prime-order transformation exists outside of Freeman's framework. Such a more efficient transformation could also provide a way to implement, e.g., the widely used Groth-Sahai proof system [63] more efficiently.

OUR CONTRIBUTION: A DIFFERENT VIEW ON COMPOSITE-ORDER-TO-PRIME-ORDER CONVERSIONS. In this work, we take a step back and question several assumptions that are implicitly made in Freeman's framework. We exhibit a different composite-order-to-prime-order conversion outside of his model, and show that it circumvents previous lower bounds. In particular, our construction leads to more efficient Groth-Sahai proofs in the symmetric setting (i.e., with a symmetric pairing). Moreover, our construction can be implemented from *any* matrix assumption [42] (including the k-linear assumption) and scales better to multilinear settings than previous approaches. In the following, we give more details on our construction and its properties.

A TECHNICAL PERSPECTIVE: A POLYNOMIAL INTERPRETATION OF LINEAR SUBSPACE. To explain our approach, recall that Freeman identifies a composite-order group with a vector space over a prime field. Moreover, in his work, subgroups of the composite-order group always correspond to *uniformly chosen* subspaces of a certain dimension. Of course, such "unstructured" subspaces only allow for rather generic interpretations of composite-order pairings (as generic "vector multiplications" as above).

Instead, we interpret the composite-order group as a very structured vector space. More concretely, we interpret a composite-order group element as (the coefficient vector of) a polynomial $f(X)$ over a prime field. In this view, a composite-order subgroup corresponds to the set of all polynomials with a common zero s (for a fixed and hidden s). Composite-order group operation and pairing correspond to polynomial addition and multiplication. Moreover, the hidden common zero s can be used as a trapdoor to decide subgroup membership, and thus to implement a "projection" in the sense of Freeman.

Specifically, our "vector multiplication" is very structured and natural, and there are several ways to implement it efficiently. For instance, we can apply a convolution on the coefficient vectors, or, more efficiently, we can represent f as a vector of evaluations $f(i)$ at sufficiently many fixed values i, and multiply these evaluation vectors component-wise. In particular, we circumvent the mentioned lower bound of Seo [107] by our different interpretation of composite-order group elements as vectors.

Another interesting property of our construction is that it scales better to the multilinear setting than previous approaches. For instance, while it seems possible to generalize at least Freeman's approach to a "projecting pairing" to a setting with a k-linear map (instead of a pairing), the corresponding generic vector multiplication would lead to exponentially (in k) large vectors in the target group. In our case, a k-linear map corresponds to the multiplication of k polynomials, and only requires a quadratic number of group elements in the target group.[1]

In the description above, f is always a univariate polynomial. With this interpretation, we can show that the SCasc assumption from Escala et al. [42] implies subgroup indistinguishability. However, we also provide a "multivariate" variant of our approach (with polynomials f in several variables) that can be implemented with *any* matrix assumption (such as the k-linear and even weaker assumptions). Furthermore, in the terminology of Freeman, we provide both a "projecting," and a "projecting and canceling" pairing construction (although the security of the "projecting and canceling" construction requires additional complexity assumptions).

APPLICATIONS. The performance improvements of our approach are perhaps best demonstrated by the case of Groth-Sahai proofs. Compared to the most efficient previous implementations of Groth-Sahai proofs in prime-order groups with symmetric pairing [108, 42], we almost halve the number of required prime-order pairing operations (cf. Table 5.1). As a bonus, we also improve on the size of prime-order group elements in the target group, while retaining the small common reference string from [42].

Additionally, we show how to implement a variant of the Boneh-Goh-Nissim encryption scheme [13] in prime-order groups with a k-linear map. As already sketched, this is possible with Freeman's approach only for logarithmically small k.

STRUCTURAL RESULTS. Of course, a natural question is whether *our* results are optimal, and if so, in what sense exactly. We can settle this question, in the following sense: we show that the construction sketched above is optimal in our generalized framework. We also prove a similar result for our construction from general matrix assumptions.

OPEN PROBLEMS. In this work, we focus on settings with a *symmetric* pairing (resp. multilinear map). It is an interesting open problem to extend our approach to asymmetric settings. Furthermore, the conversion that leads to a canceling *and* projecting map (in the terminology of Freeman) requires a nonstandard complexity assumption (that however holds generically, as we prove). It would be interesting to find constructions from more standard assumptions.

5.2 Preliminaries

INTERPOLATING SETS. Let $\vec{X} = (X_1, \ldots, X_d)$ be a vector of variables. Let $W \subset \mathbb{Z}_p[\vec{X}]$ be a subspace of polynomials of finite dimension m. Given a set of polynomials $\{\mathfrak{r}_0, \ldots, \mathfrak{r}_{m-1}\}$ which are a basis of W, we say that $\vec{x}_1, \ldots, \vec{x}_m \in \mathbb{Z}_p^d$ is an *interpolating*

[1] We multiply k polynomials, and each polynomial should be of degree at least k, in order to allow for suitable subgroup indistinguishability problems that are plausible even in face of a k-linear map.

set for W if the matrix

$$\begin{pmatrix} \mathfrak{r}_0(\vec{x}_1) & \dots & \mathfrak{r}_{m-1}(\vec{x}_1) \\ \vdots & & \vdots \\ \mathfrak{r}_0(\vec{x}_m) & \dots & \mathfrak{r}_{m-1}(\vec{x}_m) \end{pmatrix}$$

has full rank. It can be easily seen that the property of being an interpolating set is independent of the basis. Further, when p is exponential (and m and the degrees of $\mathfrak{r}_i$ are polynomial) in the security parameter, any m random vectors $\vec{x}_1, \dots, \vec{x}_m$ form an interpolating set with overwhelming probability.

5.3 Our Framework

We now present our definitional framework for composite-to-prime-order transformations. Basically, the definitions in this section will enable us to describe how groups of prime order p with a multilinear map e can be converted into groups of order p^n for some $n \in \mathbb{N}$ with a multilinear map $\tilde{e}$. These converted groups will then "mimic" certain features of composite-order groups. Since $\tilde{e}$ is just a composition of several instances of e, we will refer to e as the *basic multilinear map*. We start with an overview of the framework of Freeman ([45]), since this is the established model for such transformations. Afterwards, we describe our framework in terms of differences to the model of Freeman.

FREEMAN'S MODEL. Freeman identifies some abstract properties of bilinear composite order groups which are essential to construct some cryptographic protocols, namely subgroup indistinguishability, the projecting property and the canceling property. For Freeman, a symmetric bilinear map generator takes a bilinear group of prime order p with a pairing e and outputs some groups $\mathbb{H} \subset G, G_T$ of order p^n for some $n \in \mathbb{N}$ and a symmetric bilinear map $\tilde{e}\colon\ G \times G \to G_T$, computed via the basic pairing e. Useful instances of such generators satisfy the subgroup indistinguishability assumption, which means that it should be hard to decide membership in $\mathbb{H} \subset G$. Further, the pairing is projecting if the bilinear map generator also outputs some maps π, π_T defined respectively on G, G_T which commute with the pairing and such that $\ker \pi = \mathbb{H}$. The pairing is canceling if $\tilde{e}(\mathbb{H}, \mathbb{H}') = 0$ for some decomposition $G = \mathbb{H} \oplus \mathbb{H}'$.

INSTANTIATIONS. Further, Freeman gives several concrete instantiations in which the subgroups $\mathbb{H}$ output by the generator are sampled uniformly. More specifically, in the language of [42], the instantiations sample subgroups according to the $\mathcal{U}_{n,\ell}$ distribution. Although his model is not specifically restricted to this case, follow-up work seems to identify "Freeman's model" with this specific matrix distribution. For instance, the results of [89] on the impossibility of achieving the projecting and canceling property simultaneously or the impossibility result of Seo [107], who proves a lower bound on the size of the image of a projecting pairing, are also in this setting.

OUR MODEL. Essentially, we recover Freeman's original definitions for the symmetric setting, however with some subtle additional precisions. First, we extend his model to multilinear maps and, like Seo [107], distinguish between basic multilinear map operations (e) and multilinear map operations ($\tilde{e}$), since an important efficiency measure is how many e operations are required to compute $\tilde{e}$. The second and main

block of differences is introduced with the goal of making the model compatible with several families of matrix assumptions, yielding a useful tool to prove optimality and impossibility results. For this, we extend Freeman's model to explicitly support different families of subgroup assumptions and state clearly what the dependency relations between the different outputs of the multilinear group generator are. In Section 5.9 the advantages of the refinement of the model will become apparent.

Definition 5.3.1. *Let $k, \ell, n, r \in \mathbb{N}$ with $k > 1$ and $r \geq n > \ell$. A $(k,(r,n,\ell))$ symmetric multilinear map generator $\mathcal{G}_{k,(r,n,\ell)}$ takes as input a security parameter λ and a symmetric prime-order k-linear map generator $\mathcal{G}_k$ and outputs in probabilistic polynomial time a tuple $(\mathcal{MG}_k, \mathbb{H}, \mathsf{G}, \mathsf{G}_T, \tilde{e})$, where*

- *$\mathcal{MG}_k := (k, G, G_T, e, p, \mathcal{P}, \mathcal{P}_T) \leftarrow \mathcal{G}_k(1^\lambda)$ is a description of a prime order symmetric k-linear group,*
- *$\mathsf{G} \subset G^r$ is a subgroup of G^r with a minimal generating set of size n,*
- *$\mathbb{H} \subset \mathsf{G}$ is a subgroup of G with a minimal generating set of size ℓ,*
- *$\tilde{e}\colon \mathsf{G}^k \to \mathsf{G}_T$ is a non-degenerate k-linear map.*

We assume that elements in $\mathbb{H}, \mathsf{G}$ are represented as vectors in G^r. With this representation, it is natural to identify elements in these groups with vectors in $\mathbb{Z}_p^r$ in the usual way, via the canonical basis. Via this identification, any subgroup $\mathbb{H} \subset G^r$ spanned by $[\vec{b}_1], \ldots, [\vec{b}_\ell]$ corresponds to the subspace H of $\mathbb{Z}_p^r$ spanned by $\vec{b}_1, \ldots, \vec{b}_\ell$, and we write $\mathbb{H} = [H]$. Further, we may assume that $\mathsf{G}_T = G_T^m$ and elements of G_T are represented by m-tuples of G_T, for some fixed $m \in \mathbb{N}$, although we do not include m as a parameter of the multilinear generator.

In most constructions $n = r$, in which case we drop the index r from the definition, and we simply refer to such a generator as a $(k,(n,\ell))$ generator $\mathcal{G}_{k,(n,\ell)}$. We always assume that membership in G is easy to decide.[2] In the case where $n = r$ and $\mathsf{G} = G^r$ this is obviously the case, but otherwise we assume that the description of G includes some auxiliary information which allows to test it (like in [108], [86] and our construction of Section 5.4.3).

Definition 5.3.2 (Properties of multilinear map generators)**.** *Let $\mathcal{G}_{k,(r,n,\ell)}$ denote a $(k,(r,n,\ell))$ symmetric multilinear map generator as in Definition 5.3.1 with output $(\mathcal{MG}_k, \mathbb{H}, \mathsf{G}, \mathsf{G}_T, \tilde{e})$. We define the following properties:*

- ***Subgroup indistinguishability.*** *We say that $\mathcal{G}_{k,(r,n,\ell)}$ satisfies the* subgroup indistinguishability property *if for all PPT adversaries $\mathcal{D}$,*

$$\begin{aligned}\mathbf{Adv}_{\mathcal{G}_{k,(r,n,\ell)}}(\mathcal{D}) =& \mathbf{Pr}[\mathcal{D}(\mathcal{MG}_k, \mathbb{H}, \mathsf{G}, \mathsf{G}_T, \tilde{e}, x) = 1] - \\ & \mathbf{Pr}[\mathcal{D}(\mathcal{MG}_k, \mathbb{H}, \mathsf{G}, \mathsf{G}_T, \tilde{e}, u) = 1] = \eta(\lambda),\end{aligned}$$

  where the probability is taken over $(\mathcal{MG}_k, \mathbb{H}, \mathsf{G}, \mathsf{G}_T, \tilde{e}) \leftarrow \mathcal{G}_{k,(r,n,\ell)}(1^\lambda)$, $x \leftarrow \mathbb{H}, u \leftarrow \mathsf{G}$ and the coin tosses of the adversary $\mathcal{D}$.

[2] We note that none of the current candidate constructions of approximate multilinear maps (e.g., [51, 3]) or graded encoding systems (see Chapter 3) provide efficient algorithms for deciding group membership. This will not affect our results, but of course hinders certain applications (such as Groth-Sahai proofs).

- ***Projecting.*** *We say that* $(\mathcal{MG}_k, \mathbb{H}, \mathsf{G}, \mathsf{G}_T, \tilde{e})$ *is* projecting *if there exist two non-zero homomorphisms* $\pi\colon \mathsf{G} \to \mathsf{G}$, $\pi_T\colon \mathsf{G}_T \to \mathsf{G}_T$ *such that* $\ker \pi = \mathbb{H}$ *and* $\pi_T(\tilde{e}(x_1, \ldots, x_k)) = \tilde{e}(\pi(x_1), \ldots, \pi(x_k))$ *for any* $(x_1, \ldots, x_k) \in \mathsf{G}^k$. *For the special case* $r = n = \ell + 1$, $\mathsf{G} := G^n$ *we can equivalently define the maps* $\pi\colon G^n \to G$, $\pi_T\colon \mathsf{G}_T \to G_T$ *such that* $\ker \pi = \mathbb{H}$ *and* $\pi_T(\tilde{e}(x_1, \ldots, x_k)) = e(\pi(x_1), \ldots, \pi(x_k))$ *(matching the original definition of [63]). As usual, we say that* $\mathcal{G}_{k,(r,n,\ell)}$ *is projecting if its output is projecting with overwhelming probability.*

- ***Canceling.*** *We say that* $(\mathcal{MG}_k, \mathbb{H}_1, \mathsf{G}, \mathsf{G}_T, \tilde{e})$ *is* canceling *if there exists a decomposition* $\mathsf{G} = \mathbb{H}_1 \oplus \mathbb{H}_2$. *Note that this means that for any* $x_1 \in \mathbb{H}_{j_1}, \ldots, x_k \in \mathbb{H}_{j_k}$ *except for* $j_1 = \ldots = j_k$, *we have that* $\tilde{e}(x_1, \ldots, x_k) = 0$. *We call* $\mathcal{G}_{k,(r,n,\ell)}$ *canceling if its output is canceling with overwhelming probability.*

So far, the definitions given match those of Freeman (extended to the k-linear case) except that we explicitly define the basic k-linear group $\mathcal{MG}_k$ which is used in the construction. We will now introduce two aspects of our framework that are new compared to Freeman's model. First, we will define multilinear generators that sample subgroups according to a specific matrix assumptions. Then, we will define a property of the multilinear map $\tilde{e}$ that will be very useful to establish impossibility results and lower bounds.

Definition 5.3.3. *Let* $k, \ell, n, r \in \mathbb{N}$ *with* $k > 1$, $r \geq n > \ell$ *and* $\mathcal{D}_{n,\ell}$ *be a matrix distribution. A* $(k, (r, n, \ell), \mathcal{D}_{n,\ell})$ *multilinear map generator* $\mathcal{G}_{k,(r,n,\ell),\mathcal{D}_{n,\ell}}$ *is a* $(k, (r, n, \ell))$ *multilinear map generator which outputs a tuple* $(\mathcal{MG}_k, \mathbb{H}, \mathsf{G}, \mathsf{G}_T, \tilde{e})$ *such that the distribution of the subspaces* H *such that* $\mathbb{H} = [H]$ *equals* $\mathcal{D}_{n,\ell}$ *for any fixed choice of* $\mathcal{MG}_k$.

As usual, in the case where $r = n$, we just drop r and refer to a $(k, \mathcal{D}_{n,\ell})$ multilinear map generator $\mathcal{G}_{k,\mathcal{D}_{n,\ell}}$. We conclude our framework with a definition that enables us to distinguish generators where the multilinear map $\tilde{e}$ may or may not depend on the choice of the subgroups.

Definition 5.3.4. *We say that a* $(k, (r, n, \ell), \mathcal{D}_{n,\ell})$ *multilinear map generator with output* $(\mathcal{MG}_k, \mathbb{H}, \mathsf{G}, \mathsf{G}_T, \tilde{e})$ *as in Definition 5.3.3 defines a* fixed multilinear map *if the random variable* H *(s.t.* $\mathbb{H} = [H]$*) conditioned on* $\mathcal{MG}_k$ *and the random variable* $(\mathsf{G}, \mathsf{G}_T, \tilde{e})$ *conditioned on* $\mathcal{MG}_k$ *are independent.*

5.4 Our Constructions

All of our constructions arise from the following *polynomial point of view*: The key idea is to treat $\mathsf{G} = G^n$ as an implicit representation of some space of polynomials. Polynomial multiplication will then give us a natural multilinear map. For subspaces $\mathbb{H}^{(\vec{s})}$ that correspond to polynomials sharing a common root $\vec{s}$, this multilinear map will turn out to be projecting. We will first illustrate this idea by means of a simple concrete example where subgroup decision for $\mathbb{H}^{(\vec{s})}$ is equivalent to 2-SCasc (Section 5.4.1). Then we show that actually any polynomially induced matrix assumption gives rise to such a polynomial space and thus allows for the construction of a k-linear projecting map (Section 5.4.2). Finally, by considering G along

Table 5.1: Efficiency of different symmetric projecting k-linear maps. The size of the domain (n) and codomain (m) of $\tilde{e}$ is given as number of group elements of G and G_T, respectively. Costs are stated in terms of application of the basic map e, group operations (gop) including inversion in G/G_T, and ℓ-fold multi-exponentiations of the form $e_1[a_1] + \cdots + e_\ell[a_\ell]$ (ℓ-mexp) in G/G_T. Note that we use an evaluate-multiply-approach for the computation of $\tilde{e}$.

Construction	Ass.	Co-/Domain	Cost $\tilde{e}$	Cost π	Cost π_T
Freeman, $k = 2$ [45]	$\mathcal{U}_2$	9/3	9 e	3 3-mexp	9 9-mexp
Seo, $k = 2$ [107]	$\mathcal{U}_2$	6/3	9 e + 3 gop	3 3-mexp	6 6-mexp
This work, $k = 2$	$\mathcal{SC}_2$	**5/3**	**5 e + 22 gop**	**1 2-mexp**	**1 5-mexp**
This work, $k = 2$	$\mathcal{U}_2$	6/3	6 e + 12 3-mexp[1]	1 3-mexp	1 6-mexp
Freeman, $k > 2$	$\mathcal{U}_k$	$(k+1)^k/k+1$	$(k+1)^{k+1}$ e	$k+1$ $(k+1)$-mexp	$(k+1)^k$ $(k+1)^k$-mexp
This work, $k > 2$	$\mathcal{U}_k$	$\binom{2k}{k}/k+1$	$\binom{2k}{k}$ e + $\binom{2k}{k}$ k $(k+1)$-mexp[1]	1 $(k+1)$-mexp	1 $\binom{2k}{k}$-mexp
This work, $k > 2$	$\mathcal{SC}_k$	$k^2+1/k+1$	(k^2+1) e + (k^3+k) k-**mexp**[1]	**1 k-mexp**	**1 k^2+1-mexp**

[1]For the construction based on $\mathcal{SC}_k$, the involved exponents are relatively small, namely the biggest one is $(\lceil \frac{k^2+1}{2} \rceil)^k$. Also for $\mathcal{U}_k$, the involved exponents can usually be made small.

with the multilinear map as an implicit representation of a polynomial ring modulo some reducible polynomial, we are able to construct a multilinear map which is both projecting and canceling (see Section 5.4.3). See Table 5.1 for an overview of the characteristics of our projecting map constructions in comparison with previous work.

5.4.1 Warm-Up: A Projecting Pairing based on the 2-SCasc Assumption

Let $(k = 2, G, G_T, e, p, \mathcal{P}, \mathcal{P}_T) \leftarrow \mathsf{Setup2}(1^\lambda)$ be the output of a symmetric prime-order bilinear group generator. We set $\mathsf{G} := G^3$ and $\mathsf{G}_T := G_T^5$. For any $[\vec{f}] = ([f_0], [f_1], [f_2]) \in \mathsf{G} = G^3$, we identify $\vec{f}$ with the polynomial $f = f_0 + f_1 X + f_2 X^2 \in \mathbb{Z}_p[X]$ of degree at most 2. Similarly, any $[\vec{f}]_T \in \mathsf{G}_T$ corresponds to a polynomial of degree at most 4. Then the canonical group operation for G and G_T corresponds to polynomial addition (in the exponent), i.e., $[\vec{f}] + [\vec{g}] = [\vec{f} + \vec{g}] = [f + g]$ and $[\vec{f}]_T + [\vec{g}]_T = [f + g]_T$. Furthermore, polynomial multiplication (in the exponent) gives a map $\tilde{e}\colon \mathsf{G} \times \mathsf{G} \to \mathsf{G}_T$,

$$\tilde{e}([\vec{f}], [\vec{g}]) := \Big(\Big[\sum_{i+j=0} f_i \mathcal{P}_j \Big]_T, \ldots, \Big[\sum_{i+j=4} f_i \mathcal{P}_j \Big]_T \Big) = [f \cdot g]_T.$$

It is easy to see that $(\mathsf{G}, \mathsf{G}_T, \tilde{e})$ is again a bilinear group setting, where the group operations and the pairing $\tilde{e}$ can be efficiently computed.

A SUBGROUP DECISION PROBLEM. For some fixed $s \in \mathbb{Z}_p$ let us consider the subgroup $\mathbb{H}^{(s)} \subset \mathsf{G}$ formed by all elements $[\vec{f}] \in \mathsf{G}$ such that $\vec{f}$ viewed as polynomial f has root s, i.e., $\mathbb{H}^{(s)} = \{[f] \in \mathsf{G} \mid f(s) = 0\}$. In other words, $\mathbb{H}^{(s)}$ consists of all $[f]$ with f of the form

$$(X - s)(f_1' X + f_0'), \tag{5.1}$$

where $f_1', f_0' \in \mathbb{Z}_p$. Thus, given $[f]$ and $[s]$, the subgroup decision problem for $\mathbb{H}^{(s)} \subset \mathsf{G}$ means to decide whether f is of this form or not. Viewing Equation (5.1)

as matrix-vector multiplication, we see that this is equivalent to deciding whether $\vec{f}$ belongs to the image of the 3×2 matrix

$$\mathbf{A}(s) := \begin{pmatrix} -s & 0 \\ 1 & -s \\ 0 & 1 \end{pmatrix}. \tag{5.2}$$

Hence, our subgroup decision problem corresponds to the 2-SCasc problem (cf. Definition 2.4.4) which is hard in a generic bilinear group [42].

PROJECTIONS. Given s, we can simply define projection maps $\pi\colon \mathsf{G} \to G$ and $\pi_T\colon \mathsf{G}_T \to G_T$ by polynomial evaluation at s (in the exponent), i.e., $[\vec{f}]$ is mapped to $[f(s)]$ and $[\vec{f}]_T$ to $[f(s)]_T$. Computing π, π_T requires group operations only. Obviously, it holds that $\ker(\pi) = \mathbb{H}^{(s)}$ and $e(\pi([\vec{f_1}]), \pi([\vec{f_2}])) = \pi_T(\tilde{e}([\vec{f_1}], [\vec{f_2}]))$.

SAMPLING FROM $\mathbb{H}^{(s)}$. Given $[(-s, 1, 0)], [(0, -s, 1)] \in \mathsf{G}$, a uniform element from $\mathbb{H}^{(s)}$ can be sampled by picking $(f'_0, f'_1) \leftarrow \mathbb{Z}_p^2$ and, as with any matrix assumption, computing the matrix-vector product

$$\left[\begin{pmatrix} -s & 0 \\ 1 & -s \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} f'_0 \\ f'_1 \end{pmatrix}\right] = \left[(-sf'_0, f'_0 - sf'_1, f'_1)^{\mathrm{T}}\right].$$

Again, this can be done using the group operation only.

EFFICIENCY. Computing $\tilde{e}$ in our construction corresponds to polynomial multiplication. Although this multiplication happens in the exponent (and we are "only" given implicit representations of the polynomials), we are not forced to stick to schoolbook multiplication. We propose to follow an evaluation-multiplication-interpolation approach (using small interpolation points) where the actual interpolation step is postponed to the computation of π_T.

More precisely, so far we used coefficient representation for polynomials over G and G_T with respect to the standard basis. However, other (s-independent) bases are also possible without affecting security. For efficiency, we propose to stick to this representation for G but to use point-value representation for polynomials over G_T with respect to the fixed interpolating set $M := \{-2, -1, 0, 1, 2\}$ (cf. Section 5.2). This means we now identify a polynomial g in the target space with the vector $(g(-2), g(-1), g(0), g(1), g(2))$.

More concretely, to compute $\tilde{e}([f_1], [f_2]) = ([(f_1 f_2)(x)]_T)_{x \in M}$, we first evaluate f_1 and f_2 (in the exponent) with all $x \in M$, followed by a point-wise multiplication $([f_1(x) f_2(x)]_T)_{x \in M} = (e([f_1(x)], [f_2(x)]))_{x \in M}$. This way, $\tilde{e}$ can be computed more efficiently with only five pairings. Computing π is unchanged. To apply π_T, one first needs to obtain the coefficient representation by interpolation and then evaluate the polynomial at s. However, this can be done simultaneously and as the 1×5 matrix describing this operation can be precomputed (given s) it does not increase the computational cost much.

5.4.2 Projecting Multilinear Maps from any Matrix Assumption

In the following, we will first demonstrate that for any vector space of polynomials, the natural pairing given by polynomial multiplication is projecting for subspaces consisting of polynomials sharing a common root. We will then show that any (polynomially induced) matrix assumption can equivalently be considered as a subspace

assumption in a vector space of polynomials of this type. This way, we obtain a natural projecting multilinear map for any polynomially induced matrix assumption.

A PROJECTING MULTILINEAR MAP ON VECTOR SPACES OF POLYNOMIALS. Let $\mathcal{MG}_k := (k, G, G_T, e, p, \mathcal{P}, \mathcal{P}_T) \leftarrow \mathsf{Setup}k(1^\lambda)$ be the output of a prime-order k-linear group generator. Let $V \subset \mathbb{Z}_p[\vec{X}]$ be a vector space of polynomials of dimension n for which we fix basis $\mathfrak{q}_0, \ldots, \mathfrak{q}_{n-1}$. Then for any $[\vec{f}] \in \mathsf{G} := G^n$ we can identify the vector $\vec{f} = (f_0, \ldots, f_{n-1})$ with a polynomial $f = \sum f_i \mathfrak{q}_i \in V$. In the 2-SCasc example above, V corresponds to univariate polynomials of degree at most 2 and the basis is given by $1, X, X^2$. On V, we have a natural k-linear map given by polynomial multiplication: $\mathsf{mult}_k \colon V^k \to \mathbb{Z}_p[\vec{X}], \mathsf{mult}_k(f_1, \ldots, f_k) = f_1 \cdots f_k$. Let $W \subset \mathbb{Z}_p[\vec{X}]$ be the span of the image of mult_k and m its dimension. Then we can again fix a basis $\mathfrak{r}_0, \ldots, \mathfrak{r}_{m-1}$ of W to identify polynomials with vectors. In the 2-SCasc example above, W consists of polynomials of degree at most 4 and we chose the basis $1, X, X^2, X^3, X^4$ of W for our initial presentation. From polynomial multiplication, we then obtain a non-degenerate k-linear map

$$\tilde{e} \colon \mathsf{G}^k \to G_T^m, \tilde{e}([\vec{f_1}], \ldots, [\vec{f_k}]) = [f_1 \cdots f_k]_T.$$

Now consider a subspace $\mathbb{H}^{(\vec{s})} \in \mathsf{G}$ of the form $\mathbb{H}^{(\vec{s})} = \{[f] \in \mathsf{G} \mid f(\vec{s}) = 0\}$. It is easy to see that $\tilde{e}$ is projecting for this subspace: A projection map $\pi \colon \mathsf{G} \to G$ with $\ker(\pi) = \mathbb{H}^{(\vec{s})}$ is given by evaluation at $\vec{s}$, i.e., $\pi([\vec{f}]) = [f(\vec{s})]$. Similarly, $\pi_T \colon G_T^m \to G_T$ is defined by $\pi_T([\vec{g}]_T) = [g(\vec{s})]_T$ and by construction we have $e(\pi([\vec{f_1}]), \ldots, \pi([\vec{f_k}])) = [f_1(\vec{s}) \cdots f_k(\vec{s})]_T = [(f_1 \cdots f_k)(\vec{s})]_T = \pi_T(\tilde{e}([\vec{f_1}], \ldots, [\vec{f_k}]))$.

FROM A POLYNOMIALLY INDUCED MATRIX DISTRIBUTION TO A SPACE OF POLYNOMIALS. Now, let $\mathcal{D}_{n-1}$ be any polynomially induced matrix distribution as defined in Definition 2.4.4 and let $\mathbf{A}(\vec{X}) \in (\mathbb{Z}_p[\vec{X}])^{n \times (n-1)}$ be the polynomial matrix describing this distribution. Then we set $\mathsf{G} := G^n$ and consider the subspace $[\mathsf{Im}\, \mathbf{A}(\vec{s})]$ for some $\vec{s}$. We now show that we can identify G with a vector space V of polynomials, such that the subspace $\mathsf{Im}\, \mathbf{A}(\vec{s})$ corresponds exactly to polynomials having a root at $\vec{s}$. To this end, consider the determinant of $(\mathbf{A}(\vec{X})||\vec{F})$ as a polynomial $\mathfrak{d}$ in indeterminates $\vec{X}$ and $\vec{F}$. Since we assume that $\mathbf{A}(\vec{s})$ has generically[3] full rank a given vector $\vec{f} \in \mathbb{Z}_p^n$ belongs to the image of $\mathbf{A}(\vec{s})$ iff the determinant of the extended matrix $(\mathbf{A}(\vec{s})||\vec{f})$ is zero, i.e., $\mathfrak{d}(\vec{s}, \vec{f}) = 0$. To obtain the desired vector space V with basis $\mathfrak{q}_0, \ldots, \mathfrak{q}_{n-1}$, we consider the Laplace expansion of this determinant to write $\mathfrak{d}$ as

$$\mathfrak{d}(\vec{X}, \vec{F}) = \sum_{i=0}^{n-1} F_i \mathfrak{q}_i(\vec{X}). \tag{5.3}$$

for some polynomials $\mathfrak{q}_i(\vec{X})$ depending only on $\mathbf{A}$. For $\mathcal{SC}_2$, we have $\mathfrak{q}_i = X^i$. We note that in all cases of interest the $\mathfrak{q}_i$ are linearly independent.

Thus, we may now identify $[\vec{f}] \in \mathsf{G}$ with the implicit representation of the polynomial $f = \mathfrak{d}(\vec{X}, \vec{f}) = \sum_i f_i \mathfrak{q}_i$ and as $f(\vec{s}) = \sum_i f_i \mathfrak{q}_i(\vec{s}) = 0$ iff $\vec{f} \in \mathsf{Im}\, \mathbf{A}(\vec{s})$ we have $\mathbb{H}^{(\vec{s})} = [\mathsf{Im}\, \mathbf{A}(\vec{s})] = \{[f] \in \mathsf{G} \mid f(s) = 0\}$. Hence, we may construct a projecting k-linear map from polynomial multiplication as described in the previous paragraph.

[3] This means that $\mathbf{A}(\vec{s})$ will be full rank with overwhelming probability and this is indeed equivalent to $\mathfrak{d} \neq 0$. To simplify the exposition, we may assume that the sampling algorithm is changed to exclude $\vec{s}$ where $\mathbf{A}(\vec{s})$ does not have full rank.

Working through the construction, one can obtain explicit coordinates as follows: let W be the span of $\{\mathfrak{q}_{i_1} \cdots \mathfrak{q}_{i_k} \mid 0 \leq i_j < n\}$ and fix a basis $\mathfrak{r}_0, \ldots, \mathfrak{r}_{m-1}$ of W. This determines coefficients $\lambda_t^{(i_1,\ldots,i_k)}$ in $\mathfrak{q}_{i_1} \cdots \mathfrak{q}_{i_k} = \sum_{t=0}^{m-1} \lambda_t^{(i_1,\ldots,i_k)} \mathfrak{r}_t$.

Recall that $\tilde{e}\colon (G^n)^k \to G_T^m$ is defined as $\tilde{e}([\vec{f_1}], \ldots, [\vec{f_k}]) = [f_1 \cdots f_k]_T$, expressed as an element of G_T^m via the basis $\vec{\mathfrak{r}}$. In coordinates this reads

$$\tilde{e}([\vec{f_1}], \ldots, [\vec{f_k}]) = \Big(\sum_{j_1 \leq \ldots \leq j_k} \lambda_0^{(j_1,\ldots,j_k)} \cdot \sum_{\substack{(i_1,\ldots,i_k) \in \\ \tau(j_1,\ldots,j_k)}} e([f_{1,i_1}], \ldots, [f_{k,i_k}]), \ldots, \tag{5.4}$$

$$\sum_{j_1 \leq \ldots \leq j_k} \lambda_{m-1}^{(j_1,\ldots,j_k)} \cdot \sum_{\substack{(i_1,\ldots,i_k) \in \\ \tau(j_1,\ldots,j_k)}} e([f_{1,i_1}], \ldots, [f_{k,i_k}])\Big),$$

where $[f_{1,i_1} \cdots f_{k,i_k}]_T$ simply denotes $(f_{1,i_1} \cdots f_{k,i_k})\mathcal{P}_T$ and $\tau(j_1, \ldots, j_k)$ denotes the set of permutations of $(j_1, \ldots, j_k)$. The last optimization can be done as $\mathfrak{q}_{i_1} \cdots \mathfrak{q}_{i_k} = \mathfrak{q}_{j_1} \cdots \mathfrak{q}_{j_k}$ for $(i_1, \ldots, i_k) \in \tau(j_1, \ldots, j_k)$. For the same reason, we have $m = \binom{n+k-1}{k}$ in the worst case. In this way, the target group in our constructions is always smaller than the target group in Freeman's construction (generalized to $k \geq 2$), which is of size n^k.

The following theorem summarizes our construction and its properties:

Theorem 5.4.1. *Let $k > 1$, $n \in \mathbb{N}$, and $\mathcal{D}_{n-1}$ be a polynomially induced matrix distribution. Let $\mathcal{G}_{k,\mathcal{D}_{n-1}}$ be an algorithm that on input of a security parameter 1^λ and a symmetric prime-order k-multilinear map generator $\mathcal{G}_k$ outputs $(\mathcal{MG}_k, \mathbb{H}^{(\vec{s})}, \mathsf{G}, \mathsf{G}_T, \tilde{e})$, where*

- *$\mathcal{MG}_k := (k, G, G_T, e, p, \mathcal{P}, \mathcal{P}_T) \leftarrow \mathcal{G}_k(1^\lambda)$,*
- *$\mathsf{G} := G^n$, $\mathbb{H}^{(\vec{s})} := [\mathsf{Im}\, \mathbf{A}(\vec{s})]$, $\mathbf{A}(\vec{s}) \leftarrow \mathcal{D}_{n-1}$,*
- *$\mathsf{G}_T := G_T^m$, where m equals the dimension of*

$$W := \{\sum_{0 \leq i_1,\ldots,i_k \leq n-1} \alpha_{i_1,\ldots,i_k} \mathfrak{q}_{i_1} \cdots \mathfrak{q}_{i_k} \mid \alpha_{i_1,\ldots,i_k} \in \mathbb{Z}_p\}$$

(as vector space), and $\mathfrak{q}_0(\vec{X}), \ldots, \mathfrak{q}_{n-1}(\vec{X}) \in \mathbb{Z}_p[\vec{X}]$ are polynomials such that

$$\det(\mathbf{A}(\vec{X})||\vec{F}) = \sum_{i=0}^{n-1} F_i \mathfrak{q}_i(\vec{X})$$

for the matrix $\mathbf{A}(\vec{X})$ describing $\mathcal{D}_{n-1}$, and

- *$\tilde{e}\colon \mathsf{G}^k \to \mathsf{G}_T$ is the map defined by Equation (5.4) for a basis $\mathfrak{r}_0, \ldots, \mathfrak{r}_{m-1}$ of W.*

Then $\mathcal{G}_{k,\mathcal{D}_{n-1}}$ is a $(k, \mathcal{D}_{n-1})$ multilinear map generator. It is projecting, where the projection maps $\pi\colon \mathsf{G} \to G$ and $\pi_T\colon \mathsf{G}_T \to G_T$ defined by $\pi(\vec{f}) := \sum_{i=0}^{n-1} \mathfrak{q}_i(\vec{s})[f_i]$ and $\pi_T(\vec{g}) := \sum_{i=0}^{m-1} \mathfrak{r}_i(\vec{s})[\mathcal{P}_i]_T$ are efficiently computable given the trapdoor $\vec{s}$. Furthermore, if the $\mathcal{D}_{n-1}$ assumption holds with respect to $\mathcal{G}_k$, then subgroup indistinguishability holds with respect to $\mathcal{G}_{k,\mathcal{D}_{n-1}}$.

Example 5.4.2. *We can construct a projecting k-linear map generator satisfying subgroup indistinguishability under k-SCasc (which is hard in a k-linear generic group model). For $\mathcal{G}_{k,\mathcal{SC}_k}$, we would get $n = k + 1$ and $\mathfrak{q}_i(X) = X^i$ if k is even and $\mathfrak{q}_i(X) = -X^i$ when k is odd, where $0 \leq i \leq k$. Using the basis $\mathfrak{r}_t(X) = X^t$ for W if k is even and $\mathfrak{r}_t(X) = -X^t$ if k is odd for $0 \leq t \leq k^2$, we obtain $\lambda_t^{(i_1,\ldots,i_k)} = 1$ for $t = i_1 + \cdots + i_k$ and $\lambda_t^{(i_1,\ldots,i_k)} = 0$ else. Note that we have $m = k^2 + 1$.*

Example 5.4.3. *We can also construct a k-linear map generator from k-lin. For $\mathcal{G}_{k,\mathcal{L}_k}$, we would have $n = k + 1$, and polynomials $\mathfrak{q}_k(X_0, \ldots, X_{k-1}) = X_0 \cdots X_{k-1}$ and $\mathfrak{q}_i(X_0, \ldots, X_{k-1}) = -\prod_{j \neq i} X_j$ for $0 \leq i \leq k-1$. As a basis for W we can simply take $\{\mathfrak{q}_{j_1} \cdots \mathfrak{q}_{j_k} \mid 0 \leq j_1 \leq \ldots \leq j_k \leq k\}$ yielding $m = \binom{n+k-1}{k}$.*

Example 5.4.4. *Like Freeman, we could also construct a k-linear map generator from the $\mathcal{U}_k$ assumption. Although the polynomials $\mathfrak{q}_i(X_{1,1}, \ldots, X_{k,k+1})$, $0 \leq i \leq k$, associated to $\mathcal{G}_{k,\mathcal{U}_k}$ have a much more complex description than in the k-lin case, the image size of the resulting map is the same, namely $m = \binom{n+k-1}{k}$, because a basis of the image is also $\{\mathfrak{q}_{j_1} \cdots \mathfrak{q}_{j_k} \mid 0 \leq j_1 \leq \ldots \leq j_k \leq k\}$.*

EFFICIENCY. As in our setting any change of basis is efficiently computable, the security of our construction only depends on the vector space V (which in turn determines W), but not on the bases chosen. So we are free to choose bases that improve efficiency. We propose to follow the same approach as in Section 5.4.1: Select points $\vec{x}_0, \ldots, \vec{x}_{m-1}$ that form an interpolating set for W and represent $f \in W$ via the vector $f(\vec{x}_0), \ldots, f(\vec{x}_{m-1})$. This corresponds to choosing the basis of W consisting of polynomials $\mathfrak{r}_0, \ldots, \mathfrak{r}_{m-1} \in W$ such that $\mathfrak{r}_i(\vec{x}_j) = 1$ for $i = j$ and 0 otherwise. For the domain V, the choice is less significant and we might simply choose the $\mathfrak{q}_i$'s that the determinant polynomial gives us. Then we can compute $\tilde{e}([\vec{f}_1], \ldots, [\vec{f}_k])$ by an evaluate-multiply approach using only m applications of e. Note that the evaluation step can also be done pretty efficiently if the $\mathfrak{q}_i$'s have small coefficients (which usually is the case). For details see Section 5.7.

5.4.3 Projecting and Canceling Multilinear Maps

OVERVIEW. By considering polynomial multiplication modulo a polynomial h, which has a root at the secret s, we are able to construct a $(k, (n = \ell + 1, \ell))$ symmetric multilinear map generator with a *non-fixed* pairing that is both projecting and canceling. Our first construction relies on a $k' := k + 1$-linear prime-order map e. The one additional multiplication in the exponent is used to perform the reduction modulo h. Based on this construction, we propose another $(k, (r = 2\ell, n = \ell + 1, \ell))$ symmetric multilinear map generator that requires only a $k' = k$-linear prime-order map. The security of our constructions is based on variants of the ℓ-SCasc assumption. We need to extend ℓ-SCasc by additional given group elements to allow for reduction in the exponent, e.g., in the simplest case hints of the form $[X^i \bmod h]$ are given. In Section 5.4.3 we show that our constructions are secure for $\ell \geq k'$ in generic k'-linear groups. We note that, to the best of our knowledge, this is the first construction of a map that is both projecting and canceling and naturally generalizes to $k > 2$. In Section 5.7.2 we consider efficiency of our constructions.

First Construction: Using a $(k+1)$-linear Prime-Order Map

In order to obtain a multilinear map that is both projecting and canceling we modify our construction based on SCasc from Section 5.4.1. On a high level, our construction works as follows. We will again identify vectors from $\mathsf{G} = G^n$ with polynomials $\mathbb{Z}_p[X]$ (in the exponent) with polynomial addition as the group operation. But now,

our k-linear map will correspond to polynomial multiplication *modulo some polynomial* $h(X)$ (where $h(X)$ will depend on s). To retain the projecting property, we ensure that $h(X)$ has a root at s, so $X-s$ divides $h(X)$. The orthogonal complement to $\mathbb{H}^{(s)}$ for the canceling property will then correspond to the span of $\frac{h(X)}{X-s}$, using the fact that $(X-s)\cdot\frac{h(X)}{X-s} \mod h = 0$.

We want to point out that, since in this construction modular reduction consumes one multiplication in the exponent, to emulate a canceling k-linear map $\tilde{e}$, our first construction will require a $k'=k+1$-linear basic prime-order map e. This will be improved in the upcoming section.

Let $2 \le k < k' < n$.[4] We start with a basic symmetric k'-linear group generator $(k', G, G_T, e, p, \mathcal{P}, \mathcal{P}_T) \leftarrow \mathcal{G}_{k'}$ for groups of prime order p. The polynomial by which we reduce is chosen as follows: Fix any degree n polynomial $h'(X)$, which for efficiency we select as $h'(X) := X^n$ and set $h(X) = h'(X) - h'(s) = X^n - s^n$ for $s \xleftarrow{R} \mathbb{Z}_p$. This choice ensures that $X-s$ divides h. We set $\mathsf{G} := G^n$ and $\mathsf{G}_T := G_T^n$ and note that, with the notation from Section 5.4.1, we can identify (using coefficient representation for polynomials everywhere) these two sets with the ring $\mathbb{Z}_p[X]/(h)$, whose elements are represented by polynomials $f \in \mathbb{Z}_p[X]$ of degree at most $n-1$. We again use polynomial addition as group operation and define our composite-order k-linear map $\tilde{e} : \mathsf{G} \times \cdots \times \mathsf{G} \to \mathsf{G}_T$ by polynomial multiplication modulo the polynomial h:

$$\tilde{e}([\vec{f}_1], \ldots, [\vec{f}_k]) := [f_1 \cdots f_k \bmod h]_T.$$

This requires reducing a polynomial of degree $k(n-1)$ modulo h. To perform this, the crucial observation is that the map $g \mapsto g \bmod h$ sending a polynomial $g = \sum g_i X^i$ of degree at most $k(n-1)$ to a polynomial of degree at most $n-1$ is a *linear* map for h fixed. Viewed as a matrix, its coefficients are given by $h_{i,j}$ where $h_i(X) := X^i \bmod h = \sum_{j=0}^{n-1} h_{i,j} X^j$ for $0 \le i \le k(n-1)$. In other words, $g \bmod h = \sum_j \sum_{i=0}^{k(n-1)} g_i h_{i,j} X^j$. Combining this with the definition of polynomial multiplication, we thus may compute $\tilde{e}([f_1], \ldots, [f_k])$ as

$$\Big(\sum_{i=0}^{k(n-1)} \sum_{j_1+\ldots+j_k=i} [h_{i,0} \cdot f_{1,j_1} \cdots f_{k,j_k}]_T, \ldots, \sum_{i=0}^{k(n-1)} \sum_{j_1+\ldots+j_k=i} [h_{i,n-1} \cdot f_{1,j_1} \cdots f_{k,j_k}]_T \Big),$$

where for our choice of $h'(X) = X^n$, $h_{i,j} = 0$ if $j \neq i \bmod n$ and $h_{j+\ell n,j} = s^\ell$. The k-linear pairing $\tilde{e}$ is efficiently[5] computable with the basic $k' \ge k+1$-linear map e, provided we know the $[h_{i,j}]$ that appear here. For this reason, we need to publish the $[h_{i,j}]$ as additional "hints". For our choice of $h'(X)$, this means publishing the $\kappa = k-1$ hints $[s^n], [s^{2n}], \ldots [s^{n(k-1)}]$. It is easy to see that, for other choices of (publicly known) $h'(X)$, all $[h_{i,j}]$ can be efficiently computed from $[h'(s)], \ldots, [h'(s)^{k-1}]$ as linear functions and vice versa. Of course, publishing these hints changes the security assumption we have to make. We will show in Theorem 5.4.6 that our construction is secure in the generic k'-linear group model.

[4] Our construction works for arbitrary $n > k'$. A larger n leads to a less efficient construction, but also permits a security proof based on a weaker assumption.

[5] Note that doing it this way means computing exactly n^k basic pairings and is thus only efficient if n and k are constant. However, we stress that our construction becomes more efficient if we assume a "graded" k-linear map where we can compute intermediate results, i.e., products of less than k polynomials.

For the smallest meaningful choice $n = 4, k = 2, k' = 3$, our construction translates to

$$\tilde{e}([\vec{f}], [\vec{f'}]) = \Big([f_0 f'_0 + s^4 f_1 f'_3 + s^4 f_2 f'_2 + s^4 f_3 f'_1]_T, [f_0 f'_1 + f_1 f'_0 + s^4 f_2 f'_3 + s^4 f_3 f'_2]_T,$$
$$[f_0 f'_2 + f_1 f'_1 + f_2 f'_0 + s^4 f_3 f'_3]_T, [f_0 f'_3 + f_1 f'_2 + f_2 f'_1 + f_3 f'_0]_T\Big),$$

which can be computed with a basic 3-linear map e from $[\vec{f}], [\vec{f'}], [s^4]$ using 16 evaluations of e.

SUBGROUPS. Again, consider the subgroup $\mathbb{H}^{(s)} \subset \mathsf{G}$ formed by all elements $[\vec{f}] \in \mathsf{G}$ such that $f(s) = 0$. Note that, since $X - s$ divides h, reducing modulo h does not change whether a polynomial has a root at s. As seen before, deciding membership in $\mathbb{H}^{(s)}$ is equivalent to deciding whether an element lies in the image of an $n \times (n-1)$-matrix $\mathbf{A}(s)$ from Equation (5.2). Since the polynomials $[h_i]$ provide additional information about s, subgroup indistinguishability does not correspond to the $(n-1)$-SCasc assumption anymore, but to the new extended $(\kappa = k-1, n-1)$-SCasc assumption relative to $\mathcal{G}_{k'}$ defined below. We will later show that this assumption holds in the generic group model.

Definition 5.4.5 (Extended SCasc assumption)**.** *Let $k', n, \kappa, \in \mathbb{N}$, $k' < n$ and consider $(k', G, G_T, e, p, \mathcal{P}, \mathcal{P}_T) \leftarrow \mathcal{G}_{k'}$ for a k'-linear map generator $\mathcal{G}_{k'}$. Set $h(X) := X^n - s^n$ for $s \xleftarrow{R} \mathbb{Z}_p$. Let $\mathbf{A} \in \mathbb{Z}_p^{n \times n-1}$ be of the form* (5.2) *with $-s$ in the main diagonal and $\vec{w} \in \mathbb{Z}_p^{n-1}$, $\vec{u} \in \mathbb{Z}_p^n$. We say that the* extended $(\kappa, n-1)$-SCasc assumption *holds relative to $\mathcal{G}_{k'}$ if for all PPT algorithms $\mathcal{A}$ we have that*

$$\big|\mathbf{Pr}[\mathcal{A}([s^n], [s^{2n}], \ldots, [s^{\kappa n}], [\mathbf{A}], [\mathbf{A}\vec{w}]) = 1] -$$
$$\mathbf{Pr}[\mathcal{A}([s^n], [s^{2n}] \ldots, [s^{\kappa n}], [\mathbf{A}], [\vec{u}])) = 1]\big|$$

is negligible, where the probability is taken over the random choices of $s, h', \vec{w}, \vec{u}$.

Projecting the elements of $\mathbb{H}^{(s)}$ to 0_{G} and sampling from the subgroups works as in Section 5.4.1. This uses that $(f \bmod h)(s) = f(s)$ if $X - s$ divides h. Additionally, we have $\mathsf{G} = \mathbb{H}^{(s)} \oplus \mathbb{H}^{(s,\perp)}$, where $\mathbb{H}^{(s,\perp)} := \{[f] \in \mathsf{G} \mid \exists \alpha \in \mathbb{Z}_p[X] : f(X) = \alpha \frac{h}{X-s}\}$. Note that $h(X)$ has no double root at s with overwhelming probability, so this sum is a direct sum. It holds that $\tilde{e}([\mathcal{P}_1], \ldots, [\mathcal{P}_n]) = 0_{\mathsf{G}_T}$ if $[\mathcal{P}_i] \in \mathbb{H}^{(s)}$ and $[\mathcal{P}_j] \in \mathbb{H}^{(s,\perp)}$ for any $i \neq j$.

Altogether, we have that $\tilde{e} : \mathsf{G} \times \cdots \times \mathsf{G} \to \mathsf{G}_T$ is a symmetric projecting and canceling k-linear map, where the group operations and $\tilde{e}$ can be efficiently computed. Note that the pairing now depends on s, hence our construction is *not a fixed pairing* as defined in Definition 5.3.4. We discuss the efficiency of our construction in Section 5.7.

CHOOSING THE POLYNOMIAL $h'(X)$. In our construction, we made the choice of $h'(X) = X^n$ for reasons of efficiency. But our construction works for any fixed choice of $h'(X)$. In fact, we may even sample a random $h'(X)$ according to any distribution. If we do the latter, we may also keep $h'(X)$ secret along with s (which leads to a weaker security assumption). In any case, we may w.l.o.g. always assume that the constant coefficient of $h'(X)$ is 0, as this does not affect h.

For our construction, if $h'(X)$ is secret, we still need to ensure that all $[h_{i,j}]$ are known, so we need to publish more hints (namely a subset of the $[h_{i,j}]$ from which

all others can be computed) rather than only $[h'(s)], \ldots, [h'(s)^{k-1}]$, hurting efficiency even more. Our security proof in the generic group model supports any choice of $h'(X)$ (random or not, public or not), provided $h'(X)$ is sampled independently from s.

One issue that may appear is that for applications one might want $h(X)$ to split completely as $h(X) = (X - s_1) \cdots (X - s_n)$, as this affects the behaviour of orthogonality: In this case, one can have n non-zero vectors $[f_1], \ldots, [f_n]$ that are pairwise $\tilde{e}$-orthogonal by setting $f_i = \frac{h}{X - s_i}$. This means $\tilde{e}([f_i], [f_j], [\mathcal{P}_3], \ldots, [\mathcal{P}_{k-1}]) = [0]_T$ for all $i \neq j$ and arbitrary $\mathcal{P}_3, \ldots, \mathcal{P}_{k-1}$. For our choice of $h'(X) = X^n$, we have that $h(X) = (X - s)(X - \zeta s) \cdots (X - \zeta^{n-1} s)$ splits completely iff there exists a primitive nth root of unity $\zeta \in \mathbb{Z}_p$, i.e., $p \bmod n = 1$.

One might also directly sample $h(X) = (X - s_1) \cdots (X - s_n)$ for uniform choices of s_i. While not covered by our restriction that h be sampled as $h(X) = h'(X) - h'(s)$ with $(h'(X), s)$ independent, our security proof extends to that case, as discussed in Theorem 5.4.7.

Modification: Using a k-linear Prime-Order Map

For our previous construction of a projecting and canceling k-linear map with a non-fixed pairing, a basic $k' = k + 1$-linear prime-order map is required. We will now give a modification that only requires a k-linear prime order map. The trade off will be that our construction gives a $(k = k', (r, n, \ell = n - 1))$ multilinear map generator as in Definition 5.3.3 with $r > n$, meaning that our group G rather than $\tilde{e}$ will depend on s and is embedded in some larger space $\mathsf{G} \subset G^r$, where $\tilde{e}$ is defined on G^r in a way independent from s.

Intuitively, one additional multiplication in the exponent is needed in order to perform the reduction, i.e., multiply products $f_{1,j_1} \ldots f_{k,j_k}$ of coefficients of $f_1, \ldots, f_k$ with coefficients $h_{i,j}$ for the reduction. If for one factor, say a_{1,j_1}, we were given $[a_{1,j_1} h_{i,j}]$ rather than $[a_{1,j_1}]$, this problem would not occur. To put us in this situation, we may consider first a simple extended version of G: Let $\overline{\mathsf{G}_{\mathsf{ext}}} \subset G^r$, where $r = (\kappa + 1) \cdot n = kn$ and $\kappa = k - 1$ is the number of hints we needed in the preceding construction, be defined as [6]

$$\overline{\mathsf{G}_{\mathsf{ext}}} = \{([f], [s^n f], [s^{2n} f], \ldots, [s^{\kappa n} f]) \mid [f] \in \mathsf{G}\}.$$

Similarly consider $\overline{\mathbb{H}_{\mathsf{ext}}}^{(s)} \subset \overline{\mathsf{G}_{\mathsf{ext}}}$, defined as $\overline{\mathbb{H}_{\mathsf{ext}}}^{(s)} = \{[f], [fs^n], \ldots \in \overline{\mathsf{G}_{\mathsf{ext}}} \mid f(s) = 0\}$. This just means that whoever initially computed $[f]$ in an application, computes and sends all $[f \cdot s^{n \cdot i}]$ alongside with it. We publish $[\mathbf{A}(s)], [s^n \mathbf{A}(s)], \ldots, s^{\kappa n} \mathbf{A}(s)]$ to allow efficient sampling from $\mathbb{H}^{(s)}$. This contains $[s^n], [s^{2n}], \ldots, [s^\kappa n]$, allowing efficient sampling from G. Testing membership in G is possible knowing $[s^n]$ using only a bilinear pairing. We still have $\dim \overline{\mathsf{G}_{\mathsf{ext}}} = \dim \mathsf{G} = n$, but we redundantly use $r = (\kappa + 1)n$ base group elements to represent elements from $\overline{\mathsf{G}_{\mathsf{ext}}}$, i.e., this yields a $(k, (r = (\kappa + 1)n, n, \ell = n - 1))$ multilinear map generator as in Definition 5.3.3 with $r > n$. Our efficiently computable symmetric projecting and canceling k-linear

[6] For general public $h'(X)$, this is to be changed to $[f], [h'(s)f], \ldots [h'(s)^\kappa f]$ and for secret $h'(X)$ to a (subset of) $[f], [h_{0,0} f], \ldots, [h_{n(k-1),n-1} f]$, increasing κ.

map is

$$\tilde{e}_{\mathsf{ext}}\colon \overline{\mathsf{G}_{\mathsf{ext}}}^k \to \mathsf{G}_T \cong G^n,$$
$$\tilde{e}_{\mathsf{ext}}(([f_1],\ldots,[s^{\kappa n}f_1]),\ldots,([f_k],\ldots,[s^{\kappa n}f_k])) = [f_1\cdots f_k \bmod h]_T,$$

or, more efficiently, a k-linear map $\overline{\mathsf{G}_{\mathsf{ext}}} \times \mathsf{G}^{k-1} \to \mathsf{G}_T$.

In this construction, for every $[\vec{f}] \in \mathsf{G}$, we are provided with $[s^{in}f_j]$ for any $0 \le i \le \kappa, 0 \le j < n$. But in fact, subsets of those are already sufficient to perform the multiplication and modular reduction. Restricting to such a subset can only improve security and reduces r, so we consider as our final proposal another extended version of G, where we reduce r to $r = 2n-2$. Consider $\mathsf{G}_{\mathsf{ext}} \subset G^{2n-2}$ and similarly $\mathbb{H}^{(s)}_{\mathsf{ext}} \subset \mathsf{G}_{\mathsf{ext}}$, defined as

$$\mathsf{G}_{\mathsf{ext}} = \left\{ ([f],[s^n f_2],[s^n f_3],\ldots,[s^n f_{n-1}]) \;\middle|\; [f] \in \mathsf{G}, f = \sum_{i=0}^{n-1} f_i X^i \right\}.$$

To see that this works out, write the product $\mathcal{P} = f_1\cdots f_k$ of polynomials $f_i = \sum f_{i,j}X^j$ as $\mathcal{P} = \sum_j \mathcal{P}_j X^j$. To compute $\mathcal{P} \bmod (X^n - s^n)$, we need to multiply each $\mathcal{P}_j$ by s^{in}, where $i = \lceil \frac{j}{n} \rceil$. Each $\mathcal{P}_j$ is a sum of terms $f_{1,j_1}\cdots f_{k,j_k}$ with $\sum_\ell j_\ell = j$, where one can easily verify that for $k < n$, in each such summand, we must have at least $\lceil \frac{j}{n} \rceil$ factors with $j_\ell \ge 2$. Consequently, we can compute $[g \bmod h]$ by picking up enough s^n-factors for each summand if we are given only $[s^n f_{i,j}]$ for $j \ge 2$. This yields an efficiently computable k-linear map

$$\tilde{e}_{\mathsf{ext}}\colon \mathsf{G}_{\mathsf{ext}}{}^k \to \mathsf{G}_T \cong G^n,$$
$$\tilde{e}_{\mathsf{ext}}(([f_1],\ldots,[s^n f_{1,n-1}]),\ldots,([f_k],\ldots,[s^n f_{k,n-1}])) = [f_1\cdots f_k \bmod h]_T,$$

which is still both projecting and canceling. To allow sampling and membership testing we publish $[\mathbf{A}(s)]$ and $[s^n\mathbf{A}(s)]$. Given the concrete form of $\mathbf{A}(s)$, this means publishing $[s],[s^n],[s^{n+1}]$. Needing only a $k' = k$-linear basic map allows us to perform our construction based on (a modified version of) k-SCasc (rather than $k+1$-SCasc). The minimal interesting example with $k = 2, n = 3$ then reads as follows:

$$\tilde{e}_{\mathsf{ext}}(([f_0],[f_1],[f_2],[s^3f_2]),([f_0'],[f_1'],[f_2'],[s^3f_2'])) =$$
$$\Big([f_0f_0' + f_1(s^3f_2') + (s^3f_2)f_1']_T, [f_0f_1' + f_1f_0' + (s^3f_2)f_2']_T, [f_2f_0' + f_1f_1' + f_0f_2']_T\Big).$$

This can be computed with only a 2-linear map using 9 basic pairing operations. In general, this construction requires n^k applications of e, both for $\mathsf{G}_{\mathsf{ext}}$ and $\overline{\mathsf{G}_{\mathsf{ext}}}$.

For $\overline{\mathsf{G}_{\mathsf{ext}}}$ our security assumption changes into asking that

$$(\mathcal{MG}_k, [\mathbf{A}(s)], [s^n\mathbf{A}(s)],\ldots,[s^{\kappa n}\mathbf{A}(s)], [\mathbf{A}(s)\vec{w}],\ldots,[s^{\kappa n}\mathbf{A}(s)\vec{w}]) \quad \text{and}$$
$$(\mathcal{MG}_k, [\mathbf{A}(s)], [s^n\mathbf{A}(s)],\ldots,[s^{\kappa n}\mathbf{A}(s)], [\vec{u}],\ldots,[s^{\kappa n}\vec{u}])$$

be computationally indistinguishable for $\mathcal{MG}_k \leftarrow \mathcal{G}_k, s, \vec{w}, \vec{u}$ uniform. Note that the $[s^{in}\mathbf{A}(s)]$ given here are required to sample from $\overline{\mathsf{G}_{\mathsf{ext}}}$ and $[s^n]$ is contained in $[s^n\mathbf{A}(s)]$, which allows to test membership in $\overline{\mathsf{G}_{\mathsf{ext}}}$. The security assumption for $\mathsf{G}_{\mathsf{ext}}$ is analogous and reads that

$$(\mathcal{MG}_k, [s], [s^n], [s^{n+1}], [\mathbf{A}(s)\vec{w}], [(s^n\mathbf{A}(s)\vec{w})_2],\ldots,[(s^n\mathbf{A}(s)\vec{w})_n]) \quad \text{and}$$
$$(\mathcal{MG}_k, [s], [s^n], [s^{n+1}], [\vec{u}], [s^n u_2],\ldots,[s^n u_n])$$

be computationally indistinguishable.

Proof of Generic Security of our Projecting and Canceling Constructions

We now show that the constructions presented in this section are secure in a generic multilinear group model. Note that the following theorem also covers all our constructions where the distribution of $h'(X)$ might contain no randomness at all and the distinguisher $\mathcal{A}$ may only get a subset of the data $\{h'(X), [h'(s)], \ldots\}$ or something efficiently computable from that (like $[h_{i,j}]$), making it only harder for $\mathcal{A}$. In particular, setting $h'(X) := X^n$, it covers the Extended SCasc assumption from Definition 5.4.5.

Theorem 5.4.6. *Let $n > k$ and $\kappa \geq 0$ arbitrary but fixed. Let $\mathbf{A}(X)$ be the matrix associated to the $n-1$-SCasc assumption. Then for any algorithm $\mathcal{A}$ in the generic k-linear group model, making at most $\mathrm{poly}(\log p)$ many oracle queries, its distinguishing advantage*

$$\begin{aligned}&|\mathbf{Pr}[\mathcal{A}(p, h'(X), [h'(s)^i]_{(i=1,\ldots,\kappa)}, [\mathbf{A}(s)], [\mathbf{A}(s)\vec{w}], [h'(s)^i\mathbf{A}(s)\vec{w}]_{(i=1,\ldots,\kappa)}) = 1] -\\ &\mathbf{Pr}[\mathcal{A}(p, h'(X), [h'(s)^i]_{(i=1,\ldots,\kappa)}, [\mathbf{A}(s)], [\vec{u}], [h'(s)^i\vec{u}]_{(i=1,\ldots,\kappa)}) = 1]|\end{aligned}$$

is negligible, where $h'(X) \xleftarrow{R} \mathbb{Z}_p[X]$ of degree n is sampled according to any distribution (but independent from $s, \vec{w}, \vec{u}$), $s \leftarrow \mathbb{Z}_p, \vec{w} \leftarrow \mathbb{Z}_p^{n-1}, \vec{u} \leftarrow \mathbb{Z}_p^n$ uniform.

Proof. W.l.o.g. we may assume $k = n - 1$. We may further assume that $h'(X)$ is a fixed, public polynomial, containing no randomness: Clearly, the distinguishing advantage η of $\mathcal{A}$ is the expected value (over the choice of $h'(X)$) of the conditional advantage $\mathbf{E}[\eta|h']$. Our argument will show that $\mathcal{A}$'s advantage for $h'(X)$ fixed is bounded by some negligible function, where the bound does not depend on $h'(X)$. This effectively means that we consider adversaries that may even depend non-uniformly on $h'(X)$.

Let us first consider the case where the distributions, which we want to show to be indistinguishable, are given by

$$(p, [h'(s)], [s], [\mathbf{A}(s)\vec{w}]) \text{ and } (p, [h'(s)], [s], [\vec{u}]), \text{ respectively.}$$

The general case will follow as a by-product of our proof, as we will (almost) pretend that multiplying by $[h'(s)]$ is for free and does not consume a pairing, so $\mathcal{A}$ can compute the missing data itself. $h'(X)$ is a public constant and the entries of $[\mathbf{A}(s)]$ are either $[0], [1]$ (which we assume to be given as part of the group's description / oracle) or $[s]$.

Following [42], this implies that the assumption we are about to prove is polynomially induced (i.e., the inputs to the adversary are obtained by evaluating bounded-degree polynomials in uniformly chosen unknowns). Consider the ideals

$$\begin{aligned}I_{\text{subgroup}} &= \left(H - h'(S), S - X, \vec{Z} - A(S)\vec{W}\right) \in \mathbb{Z}_p[H, X, S, \vec{W}, \vec{Z}] \text{ and}\\ I_{\text{uniform}} &= \left(H - h'(S), S - X\right) \in \mathbb{Z}_p[H, X, S, \vec{W}, \vec{Z}].\end{aligned}$$

Here, $\vec{Z} - A(S)\vec{W}$ is shorthand for the n polynomial relations of a matrix-vector product, where $A(S)$ is the $n \times n-1$ matrix of the SCasc assumption with polynomials as entries, so $A_{i,i} = -S, A_{i,i+1} = 1$ and $A_{i,j} = 0$ for $j \notin \{i, i+1\}$.

The H-variable corresponds to the hint that makes this different from the non-extended SCasc assumption, the X-variable corresponds to the known entries of the matrix A, the Z-variables to either $[A\vec{w}]$ or $[\vec{u}]$ and $\vec{W}, S$ are the uniformly chosen unknowns. $\vec{W}, S$ are only accessible to the adversary via the relations from the ideals. Note that these two ideals encode all relationships between these data.

Now, consider the ideals $J_{\text{subgroup}} = I_{\text{subgroup}} \cap \mathbb{Z}_p[H, X, \vec{Z}], J_{\text{uniform}} = I_{\text{uniform}} \cap \mathbb{Z}_p[H, X, \vec{Z}]$, which encode the relations in those variables $(H, X, \vec{Z})$ that the adversary sees. By [42, Theorem 3] (which was only proven for matrix assumptions, but the statement and proof extend directly to our setup), it suffices to show that $J_{\text{subgroup},\leq n-1} = J_{\text{uniform},\leq n-1}$, the subscripts denoting restriction to total degree $\leq n-1$.

As mentioned briefly above we will strengthen the adversary and allow it to compute polynomials $p(H, \vec{Z}, X)$ of degree totaling at most $k = n-1$ in $(\vec{Z}, X)$, i.e., we lift any degree restrictions on H. This means showing $J_{\text{subgroup},(\vec{Z},\,X)\text{-degree}\,\leq n-1} = J_{\text{uniform},(\vec{Z},\,X)\text{-degree}\,\leq n-1}$.

Translated back from the language of ideals to generic algorithms, this corresponds to allowing the adversary to multiply by $h'(S)$ for free (provided the degrees of all polynomials appearing remain bounded), thereby allowing it to compute the missing data. As a side remark, the bound κ (which gives a restriction on how $\mathcal{A}$ is allowed to multiply by $h'(S)$) is required, because otherwise equality of those ideals, restricted by degrees, no longer is equivalent to generic security (and hence the translation back from the language of ideals to generic algorithms fails). Working through [42, Theorem 3] gives a bound on the distinguishing advantage via the Schwarz-Zippel lemma, which depends on the maximal degree of any polynomial that can appear. To ensure this bound is negligible, we need κ to be constant (and the bound is uniform in $h'(X)$). Still, we can forget about κ in our proof here from now on.

To compute J_{subgroup} and J_{uniform} from I_{subgroup} and I_{uniform}, we need to eliminate the S and $\vec{W}$-variables. Elimination of S means just using $X - S$ to plug in X for S. Elimination of the $\vec{W}$-variables can be done as in the security proof of the non-extended SCasc (the additional hint H does not affect that part of the proof), so we have

$$J_{\text{subgroup}} = \Big(H - h'(X), \mathfrak{d}(Z, X)\Big) \text{ and}$$
$$J_{\text{uniform}} = \Big(H - h'(X)\Big).$$

where $\mathfrak{d}(Z, X) = \pm Z_0 \pm Z_1 X \pm \ldots \pm Z_{n-1} X^{n-1}$ is the determinant polynomial of SCasc for some specific choice of signs. It was shown in [42] to be absolutely irreducible.

Of course, $J_{\text{uniform}} \subset J_{\text{subgroup}}$. So, assume towards a contradiction that there exists some adversarially computable polynomial $\mathfrak{p}(H, \vec{Z}, X) \in J_{\text{subgroup}} \setminus J_{\text{uniform}}$ of total degree in $\vec{Z}, X$ at most $k = n-1$. By definition this implies that there exist polynomials $\mathfrak{a}, \mathfrak{b} \in \mathbb{Z}_p[H, \vec{Z}, X]$ such that

$$\mathfrak{p}(H, \vec{Z}, X) = \mathfrak{a}(H, \vec{Z}, X) \cdot \mathfrak{d}(\vec{Z}, X) + \mathfrak{b}(H, \vec{Z}, X) \cdot (H - h'(X)). \tag{5.5}$$

The existence of $\mathfrak{b}$ in the above equation (for $\mathfrak{p}, \mathfrak{a}$ fixed) is equivalent to just plugging

in $h'(X)$ for any occurrence of H, so we have

$$\mathfrak{p}'(\vec{Z}, X) := \mathfrak{p}(h'(X), \vec{Z}, X) = \mathfrak{a}(h'(X), \vec{Z}, X) \cdot \mathfrak{d}(\vec{Z}, X) = \mathfrak{a}'(\vec{Z}, X) \cdot \mathfrak{d}(\vec{Z}, X), \quad (5.6)$$

where $\mathfrak{a}'(\vec{Z}, X) := \mathfrak{a}(h'(X), \vec{Z}, X) \neq 0 \in \mathbb{Z}_p[\vec{Z}, X]$, as otherwise $\mathfrak{p} \in J_{\text{uniform}}$.

Let us give some intuition what we need to show here. The theorem from [42] essentially says that the only thing the adversary can do if the determinant $\mathfrak{d}$ is irreducible is to compute this determinant or a multiple thereof. This remains true in our case. For the usual SCasc assumption this was easily shown to be impossible, because the determinant had a higher degree than anything the adversary could compute. In our case, the situation changes, because the adversary has the polynomial H, which corresponds to S^n, at its disposal and S^n has the same degree as $\mathfrak{d}$. It is actually still easy to show that the adversary can not compute $\mathfrak{d}$ itself. The real problem is to show that this also holds for multiples $\mathfrak{a}(h'(X), \vec{Z}, X) \cdot \mathfrak{d}(\vec{Z}, X)$.

To prove our theorem, we will show that indeed $\mathfrak{a}' = 0$ is the only solution of Eq. (5.6), even when extending the base field to an algebraic closure $\overline{\mathbb{Z}_p}$.
Let us make another assumption simplifying the proof: Changing $h'(X)$ into $h'(X)+c$ for any constant $c \in \overline{\mathbb{Z}_p}$ does not affect the statement of the theorem. By using such a change, we may assume that $h'(X)$ is square-free. Note here that the condition that $h'(X) + c$ be square-free is equivalent to requiring the discriminant of $h'(X) + c \neq 0$. The discriminant of $h'(X) + c$ is a polynomial in c, which equals the resultant $\mathrm{Res}_X(\frac{\mathrm{d}h'}{\mathrm{d}X}, h'(X) + c)$ up to some normalization constant. Computing the determinant of the Sylvester matrix for this resultant results in a leading term of $n^n c^{n-1}$, so the discriminant does not vanish identically and we can find a value c (in the base field, even, if $n \geq p$) such that $h'(X) + c$ is square-free.

Let $a_0, \ldots, a_{n-1}$ be the n distinct roots of $h'(X)$ in $\overline{\mathbb{Z}_p}$. After performing a linear, invertible change of variables (which does not affect anything at hand here), we may consider the variables $\vec{Z}'$ instead of $\vec{Z}$, defined by

$$Z'_i = \mathfrak{d}(\vec{Z}, a_i) = \sum_j \pm Z_j \cdot a_i^j$$

and express everything in terms of $\vec{Z}'$, redefining $\mathfrak{p}, \mathfrak{p}', \mathfrak{a}, \mathfrak{a}', \mathfrak{d}$ accordingly as if we had made h' square-free and expressed everything in terms of $\vec{Z}'$ from the beginning. This will simplify things later, as now $\mathfrak{d}(\vec{Z}', a_i) = Z'_i$. Note here that the matrix of the linear map relating $\vec{Z}'$ and $\vec{Z}$ is a Vandermonde matrix and hence invertible.

Our proof will proceed in two steps. In the first step, we will show that if h' divides $\mathfrak{p}'$ (which corresponds to $\mathfrak{p}(H, \vec{Z}, X)$ being a multiple of H), then we can divide everything by H to obtain another non-trivial solution of Equation (5.6) with smaller degrees. In the second step, we will show that it is always the case the h' divides $\mathfrak{p}'$. This leads to a contradiction.

For the first step, let us consider the case where h' divides $\mathfrak{p}'$ and consequently h' divides $\mathfrak{a}' \cdot \mathfrak{d}$. Since $\overline{\mathbb{Z}_p}[H, \vec{Z}', X]$ is factorial, $\mathfrak{d}(\vec{Z}', X)$ is absolutely irreducible and h' can't divide $\mathfrak{d}$ for degree reasons, this means that h' must divide $\mathfrak{a}'$. In this case, we may divide both $\mathfrak{p}'$ and $\mathfrak{a}'$ by h' to obtain another solution $(\widetilde{\mathfrak{p}}', \widetilde{\mathfrak{a}}')$ of (5.6) with $\mathfrak{a}' = \widetilde{\mathfrak{a}}' \cdot h', \mathfrak{p}' = \widetilde{\mathfrak{p}}' \cdot h'$. Note that we can uniquely recover $\mathfrak{p}$ from $\mathfrak{p}'$ due to the degree restriction: For any polynomial $\mathfrak{f} \in \overline{\mathbb{Z}_p}[\vec{Z}', X]$, let $C(\mathfrak{f}) \in \overline{\mathbb{Z}_p}[H, \vec{Z}', X]$ be the *unique* polynomial of degree at most $n - 1$ in X, such that $C(\mathfrak{f})(h'(X), \vec{Z}', X) = \mathfrak{f}(\vec{Z}', X)$.

By uniqueness, we have $H \cdot C(\widetilde{\mathfrak{p}}') = C(h' \cdot \widetilde{\mathfrak{p}}')$. Consequently, $\mathfrak{p} = C(\mathfrak{p}') = C(h' \cdot \widetilde{\mathfrak{p}}') = H \cdot C(\widetilde{\mathfrak{p}}')$. This means that H divides $\mathfrak{p}$ and we may divide $\mathfrak{p}$ by H to obtain $\widetilde{\mathfrak{p}}$, such that $(\widetilde{\mathfrak{p}}, C(\widetilde{\mathfrak{a}}'))$ is another solution of (5.5). Note that by construction $\widetilde{\mathfrak{p}}$ still satisfies the degree restrictions and $\widetilde{\mathfrak{a}}' \neq 0$. After performing this transformation from $\mathfrak{p}$ to $\widetilde{\mathfrak{p}}$ finitely many times, we are in the case where h' does not divide $\mathfrak{p}'$, so we may w.l.o.g. assume from now on that h' does not divide $\mathfrak{p}'$.

We now show in the second step that h' divides $\mathfrak{p}'$, leading to a contradiction. For this, we take Equation (5.6) modulo h'

$$\mathfrak{p}''(\vec{Z}', X) := \mathfrak{p}'(\vec{Z}', X) \bmod h' = \mathfrak{p}(0, \vec{Z}', X) = (\mathfrak{a}'(\vec{Z}', X) \cdot \mathfrak{d}(\vec{Z}', X)) \bmod h'.$$

The degree restrictions on $\mathfrak{p}$ imply that $\mathfrak{p}''$ is now a polynomial of total degree at most $n-1$.

Let us plug in a_i for X in both sides of this equation. Since a_i was defined as a root of h' we have $(\mathfrak{f} \bmod h')(a_i) = \mathfrak{f}(a_i)$ and by definition of the Z_i''s in terms of Z_i, we obtain:

$$\mathfrak{p}''(\vec{Z}', a_i) = \mathfrak{a}'(\vec{Z}', a_i) \cdot \mathfrak{d}(\vec{Z}', a_i) = \mathfrak{a}'(\vec{Z}', a_i) \cdot Z_i', \qquad \text{for all } 0 \leq i \leq n-1.$$

Now consider the coefficient $c_{\vec{\alpha}} \in \overline{\mathbb{Z}_p}[X]$ of $\vec{Z}'^{\vec{\alpha}}$ in $\mathfrak{p}''(\vec{Z}', X)$. Since $\mathfrak{p}''(\vec{Z}', a_i)$ is divisible by Z_i', we must have $c_{\vec{\alpha}}(a_i) = 0$ whenever $\alpha_i = 0$. If $|\alpha|_1 = \gamma$, there are at least $n - \gamma$ indices i, such that $\alpha_i = 0$. Consequently, $c_{\vec{\alpha}}$ has at least $n - \gamma$ distinct roots. But our degree restriction on $\mathfrak{p}''$ means that $c_{\vec{\alpha}}$ can have degree at most $n - 1 - \gamma$. Hence all $c_{\vec{\alpha}}$ are 0 and $\mathfrak{p}'' = 0$. This in turn means that h' divides $\mathfrak{p}'$, which we ruled out above, giving us a contradiction. This shows that such a $\mathfrak{p}$ can't exist, finally finishing the proof.

□

Generic Security For h Composed of Random Linear Factors

In our first construction of a projecting and canceling pairing above, we discussed that it might be desirable to choose h as $h(X) = (X - s_1) \cdots (X - s_n)$, where s_1 corresponds to s. This is not of the form $h(X) = h'(X) - h'(s_1)$ where $h'(X)$ is sampled independently from $s = s_1$ and hence our proof above does not directly apply to this case. However, the case $h(X) = (X - s_1) \cdots (X - s_n)$ is essentially equivalent to setting $h'(X)$ uniform, conditioned on the event that $h'(X) - h'(s_1)$ splits completely over the base field. Intuitively, we expect that conditioning on the event that $h'(X) - h'(s_1)$ splits completely can not change generic security. The reason is that generic security can be expressed as an equality of ideals up to some degree as in [42] or by the Uber-Assumption Theorem from [11, 22]. In any case, it boils down to a problem of linear algebra, which does not depend on whether we are in $\mathbb{Z}_p$ or in the algebraic closure $\overline{\mathbb{Z}_p}$ and in the latter case, every polynomial splits completely. Rather than making this precise, we will show the stronger statement that security of choosing $h = (X - s_1) \cdots (X - s_n)$ is implied by security of choosing $h = h'(X) - h'(s_1)$, h' uniform *in the standard model*.

Theorem 5.4.7. *Let $n > k$ and let $\mathcal{G}_k$ be a symmetric prime-order k-linear group generator. Consider a PPT adversary $\mathcal{A}$ with advantage*

$$\eta = \big|\mathbf{Pr}\big[\mathcal{A}(\mathcal{MG}_k, [h_{0,0}], \ldots, [h_{k(n-1),n-1}], [\mathbf{A}], [\mathbf{A}\vec{w}]) = 1\big] - \mathbf{Pr}\big[\mathcal{A}(\mathcal{MG}_k, [h_{0,0}], \ldots, [h_{k(n-1),n-1}], [\mathbf{A}], [\vec{u}]) = 1\big]\big|,$$

where $\mathcal{MG}_k := (k, G, G_T, e, p, \mathcal{P}, \mathcal{P}_T) \leftarrow \mathcal{G}_k(1^\lambda)$*,* $s_1, \ldots, s_n \in \mathbb{Z}_p, \vec{u} \in \mathbb{Z}_p^n, \vec{w} \in \mathbb{Z}_p^{n-1}$ *uniform,* $h(X) = (X - s_1) \cdots (X - s_n)$ *and* $h_{i,j}$ *is the* j*th coefficient of* $X^i \bmod h$*. Assume* $\eta > \frac{1}{\mathrm{poly}(\lambda)}$*.*

Then there exists another PPT adversary $\mathcal{A}'$ *with*

$$\eta' = |\mathbf{Pr}[\mathcal{A}'(\mathcal{MG}_k, [h_{0,0}], \ldots, [h_{k(n-1),n-1}], [\mathbf{A}], [\mathbf{A}\vec{w}]) = 1] - \mathbf{Pr}[\mathcal{A}'(\mathcal{MG}_k, [h_{0,0}], \ldots, [h_{k(n-1),n-1}], [\mathbf{A}], [\vec{u}]) = 1]| > \eta(\lambda),$$

where $\mathcal{MG}_k := (k, G, G_T, e, p, \mathcal{P}, \mathcal{P}_T) \leftarrow \mathcal{G}_k(1^\lambda)$*,* $s_1 \in \mathbb{Z}_p, \vec{u} \in \mathbb{Z}_p^n, \vec{w} \in \mathbb{Z}_p^{n-1}$ *uniform,* $h'(X) \leftarrow \mathbb{Z}_p[X]$ *is a uniformly chosen polynomial of degree* n *with leading coefficient 1 and constant coefficient 0,* $h(X) = h'(X) - h'(s_1)$ *and* $h_{i,j}$ *is the* j*th coefficient of* $X^i \bmod h$*.*

Proof. With overwhelming probability, the s_i in the first variant are pairwise different and in the second variant $h = h'(X) - h'(s_1)$ is square-free. So it is sufficient to consider the (statistically close) variants, where we sample $(s_1, \ldots, s_n)$ uniform, conditioned on being pairwise different, respectively (h', s_1) uniform, conditioned on $h'(X) - h'(s_1)$ square-free. For $s_1, \ldots, s_n$ pairwise different, the map

$$(s_1, \ldots, s_n) \mapsto (s_1, h' = (X - s_1) \cdots (X - s_n) - s_1 \cdots s_n)$$

is exactly $(n-1)!$ to 1. As a consequence, if we sample $(h'(X), s_1)$, conditioned on the event $\mathtt{Split}$ that $h'(X) - h'(s_1)$ completely splits over $\mathbb{Z}_p$, we obtain exactly the same distribution on h as if we had sampled h as $h(X) = (X - s_1) \cdots (X - s_n)$. Further, we have $\mathbf{Pr}[\mathtt{Split}] = \frac{1}{(n-1)!}$. By a standard argument, there exists at least an $\frac{\eta}{2}$-fraction of "good" choices of h in the first variant, where the advantage of $\mathcal{A}$, conditioned on this h, is at least $\frac{\eta}{2}$.

As a consequence, simply running $\mathcal{A}$ on the second variant will give us a conditional advantage of at least $\frac{\eta}{2}$ for at least a $\frac{\eta}{2 \cdot (n-1)!}$ - fraction of "good" choices of h. For other values of h, we can simply guess to obtain an advantage of 0. Unfortunately, we cannot easily detect whether we have a good h. However, we can define $\mathcal{A}'$ as follows: First run a statistical test, which outputs 1 with overwhelming probability if the conditional advantage of $\mathcal{A}$ for the given h is at least $\frac{\eta}{4}$ and outputs 0 with overwhelming probability, if the conditional advantage of $\mathcal{A}$ is at most $\frac{\eta}{8}$. If this test outputs 1, $\mathcal{A}'$ can simply use $\mathcal{A}$ to output its final answer, otherwise $\mathcal{A}'$ just guesses. Note that since $\eta > \frac{1}{\mathrm{poly}(\lambda)}$, such a statistical test can be performed in probabilistic polynomial time, using the fact that $\mathcal{A}'$ can create instances for given $\mathcal{MG}_k, [h_{i,j}], [\mathbf{A}]$ itself by sampling its own $\vec{w}$'s respectively $\vec{u}$'s. Also, note that this reduction is not black-box, because the code of $\mathcal{A}'$ depends on the advantage η. □

5.5 The Polynomial Viewpoint

In the construction of our projecting multilinear map in Section 5.4.2, we claimed that the $\mathfrak{q}_i$'s we obtained there were linearly independent for all interesting matrix assumptions. We will make this more precise now, saying what the uninteresting matrix assumptions are here: For this, let $\mathbf{A} \in (\mathbb{Z}_p[\vec{X}])^{n \times (n-1)}$ be a matrix describing a generically full rank, polynomially induced matrix assumption. This gives us subspaces $\mathbb{H}^{(\vec{s})} \subset G^n$ with $\mathbb{H}^{(s)} = [\mathsf{Im}\, \mathbf{A}(\vec{s})]$. Consider the case where for any fixed

value of $\vec{s}$, for $\vec{w} \in \mathbb{Z}_p^n$ uniform, the distribution of one of the components, say the last one, of $\mathbf{A}(\vec{s})\vec{w}$ is uniform and independent from the other components. This last component then has no bearing whatsoever on the hardness of distinguishing $([\mathbf{A}(\vec{s})], [\mathbf{A}(\vec{s})\vec{w}])$ from $([\mathbf{A}(\vec{s})], [\vec{u}])$ for $[\vec{u}]$ uniform and we might just as well drop it. Slightly more generally, consider the following definition:

Definition 5.5.1. *Let $\mathbf{A} \in (\mathbb{Z}_p[\vec{X}])^{n\times(n-1)}$ be a matrix describing a (generically) full rank, polynomially induced matrix distribution as above. We call $\mathbf{A}$ or its associated matrix distribution* redundant*, if there exists a matrix $\mathbf{B} \in \mathbb{Z}_p^{n\times n}$, independent from $\vec{s}$, such that for all fixed $\vec{s}$ the last component of $\mathbf{B} \cdot \mathbf{A}(\vec{s})\vec{w}$ is uniform and independent from the other components over a uniformly random choice of $\vec{w}$.*

Even if the $\mathfrak{q}_i$'s are not linearly independent, we can still view elements as polynomials as follows: as in Section 5.4.2, consider the determinant polynomial $\mathfrak{d} = \det(\mathbf{A}(\vec{X})||\vec{F})$ as a polynomial in $\vec{X}, \vec{F}$ and let

$$\mathfrak{d} = \sum \mathfrak{q}_i(\vec{X}) \cdot F_i$$

be its Laplace expansion. So the $\mathfrak{q}_i$'s are (up to sign) the determinants of the $(n-1)\times(n-1)$-minors of $\mathbf{A}$. Let $V \subset \mathbb{Z}_p[\vec{X}]$ be their span. Even if the $\mathfrak{q}_i$'s may be linearly dependent, we can still map vectors to polynomials as we did before. For any $[\vec{f}] \in G^n$, we can consider the polynomial $\sum f_i\mathfrak{q}_i$. This means we have a surjective map

$$\Phi\colon\ G^n \cong \mathsf{G} \to V, \Phi([\vec{f}]) = \sum_i f_i\mathfrak{q}_i(\vec{X})$$

realizing the polynomial viewpoint.

Theorem 5.5.2. *Let $\mathbf{A} \in (\mathbb{Z}_p[\vec{X}])^{n\times(n-1)}$ be a matrix describing a generically full rank, polynomially induced matrix distribution, which is not redundant. Then Φ is bijective.*

Proof. Consider the case where Φ is not injective. Then there exist a non-zero vector $[\vec{v}] \in \ker\Phi$. By definition, $\Phi([\vec{v}])(\vec{s}) = \sum_i \mathfrak{q}_i(\vec{s})v_i = 0 = \mathfrak{d}(\vec{s}, \vec{v})$ for all $\vec{s}$. So actually, $[\vec{v}] \in \mathbb{H}^{(\vec{s})} = [\mathsf{Im}\,\mathbf{A}(\vec{s})]$ for all $\vec{s}$. Let $\mathbf{B}$ be some invertible matrix such that $\mathbf{B}\vec{v}$ is the last unit vector. Then $(0,\ldots,0,1) \in \mathsf{Im}\,\mathbf{B}\mathbf{A}(\vec{s})$, hence the last component from a random element from this image is uniform and independent from the other components, contradicting that $\mathbf{A}$ is not redundant. □

We remark that, by setting $\tilde{e}([\vec{f_1}],\ldots,[\vec{f_k}]) := [\Phi(\vec{f_1})\cdots\Phi(\vec{f_k})]_T$, we can define a projecting multilinear map either way. If Φ is not injective, the effect of Φ is exactly to drop any redundant components. The only place where we need that Φ is injective is for the lower bounds in our optimality proof in Section 5.6.2. Of course, with redundant matrix assumptions, one can beat this lower bound as follows: Take a projecting multilinear map for a $\mathcal{D}_{n,n-1}$ matrix assumption with image size m and artificially increase n by redundant components (this corresponds to adding an identity matrix block with $\mathbf{A}$ becoming block diagonal) and have the multilinear map ignore them (which is what our map does).

5.6 Optimality and Impossibility Results

In this section we consider various flavors of optimality of our construction. Additionally, we elaborate on an existing result from the literature regarding the non-existence of a projecting and canceling pairing.

5.6.1 Optimality of Polynomial Multiplication

First, we show that for any polynomially induced matrix assumption $\mathcal{D}_{\ell+1,\ell}$, the projecting multilinear map resulting from the polynomial viewpoint is optimal in terms of image size.

Theorem 5.6.1. *Let $k > 0$, and let $\mathcal{D}_{\ell+1,\ell}$ be a polynomially induced matrix assumption and let $\mathfrak{q}_0, \ldots, \mathfrak{q}_\ell$ be the polynomials associated to $\mathcal{D}_{\ell+1,\ell}$ as defined in Equation* (5.3) *in Section 5.4.2 and let $W \subset \mathbb{Z}_p[\vec{X}]$ be the space of polynomials spanned by $\{\mathfrak{q}_{i_1} \ldots \mathfrak{q}_{i_k} \mid 0 \leq i_j \leq \ell\}$. Let $(\mathcal{MG}_k, \mathbb{H}, G^{\ell+1}, G_T^m, \tilde{e})$ be the output of any other fixed $(k, \mathcal{D}_{\ell+1,\ell})$ projecting multilinear map generator. Then, $\overline{m} := \dim W \leq m$.*

$$\begin{array}{ccc} \mathsf{G}^k & \xrightarrow{\tilde{e}} & G_T^m \\ \downarrow{\scriptstyle \left(\pi^{(\vec{s})}\right)^k} & & \downarrow{\scriptstyle \pi_T^{(\vec{s})}} \\ G^k & \xrightarrow{e} & G_T \end{array} \qquad \begin{array}{ccc} \mathsf{G}^k \times \ldots \times \mathsf{G}^k & \xrightarrow{(\tilde{e}, \ldots, \tilde{e})} & G_T^m \times \ldots \times G_T^m \\ \downarrow{\scriptstyle \left(\left(\pi^{(\vec{s}_1)}\right)^k, \ldots, \left(\pi^{(\vec{s}_{\overline{m}})}\right)^k\right)} & & \downarrow{\scriptstyle \left(\pi_T^{(\vec{s}_1)}, \ldots, \pi_T^{(\vec{s}_{\overline{m}})}\right)} \\ G^k \times \ldots \times G^k & \xrightarrow{(e, \ldots, e)} & G_T \times \ldots \times G_T \end{array}$$

Figure 5.1: Left: Projecting property. Right: The diagram repeated $\overline{m}$ times for an interpolating set $\vec{s}_1, \ldots \vec{s}_{\overline{m}}$ for W.

PROOF INTUITION. The first part of the proof shows that w.l.o.g. we can assume that $\pi_T^{(\vec{s})} \circ \tilde{e}$ is polynomial multiplication for all $\vec{s}$, that is, for any $[\vec{f_1}], \ldots, [\vec{f_k}] \in G^{\ell+1}$, $\pi_T(\tilde{e}([\vec{f_1}], \ldots, [\vec{f_k}])) = [(f_1 \ldots f_k)(\vec{s})]_T$. This follows from the commutative diagram on the left, i.e., the projecting property, together with the fact that, because $\mathbb{H}$ has co-dimension 1, the map $\pi^{(\vec{s})}$ must (up to scalar multiples) correspond to polynomial evaluation at $\vec{s}$. The intuition for the second part of the proof is given by Figure 5.1. Here we show that if $\vec{s}_1, \ldots \vec{s}_{\overline{m}}$ is an interpolating set for W, then the span of $\left\{(\pi_T^{(\vec{s}_1)}(\vec{x}), \ldots, \pi_T^{(\vec{s}_{\overline{m}})}(\vec{x})) | \vec{x} \in \tilde{e}(\mathsf{G}^k)\right\} \subset G_T^{\overline{m}}$ is of dimension $\overline{m}$. This dimension can be at most the dimension of the span of $\tilde{e}(\mathsf{G}^k)$, showing $\overline{m} \leq m$.

We now give the formal proof of Theorem 5.6.1

Proof. By assumption, $\mathbb{H} = \mathbb{H}^{(\vec{s})}$ is the subspace of $\mathsf{G} = G^{\ell+1}$ spanned by the rows of the matrix $[\mathbf{A}(\vec{s})]$, for some $\vec{s} \in \mathbb{Z}_p^d$, and by definition of $\mathfrak{q}_0, \ldots, \mathfrak{q}_\ell$, if $[\mathbf{A}(\vec{s})]$ has full rank, $\mathbb{H}^{(\vec{s})} = \{\vec{f} = (f_0, \ldots, f_\ell) \mid \sum_{i=0}^{\ell} f_i \mathfrak{q}_i(\vec{s}) = 0\}$.

The fact that the map is projecting (cf. Definition 5.3.2 or Figure 5.2) guarantees that for every $\mathbb{H}^{(\vec{s})}$ there exist $\pi^{(\vec{s})} \colon \mathsf{G} \to G$, and $\pi_T^{(\vec{s})} \colon \mathsf{G}_T \to G_T$, such that $\ker \pi^{(\vec{s})} = \mathbb{H}^{(\vec{s})}$ and $e(\pi^{(\vec{s})}(\vec{x}_1), \ldots, \pi^{(\vec{s})}(\vec{x}_k)) = \pi_T^{(\vec{s})}(\tilde{e}(\vec{x}_1, \ldots, \vec{x}_k))$ for any $\vec{x}_1, \ldots, \vec{x}_k$, where e is the basic pairing operation in $\mathcal{MG}_k$. We stress that $\pi^{(\vec{s})}$ and $\pi_T^{(\vec{s})}$ may depend on $\mathbb{H}$, while by assumption, the multilinear map $\tilde{e}$ is fixed and thus independent of $\mathbb{H}$.

We structure the proof into two steps: The first step is a lemma, which says that $\pi^{(\vec{s})}$ can be viewed as polynomial evaluation at $\vec{s}$.

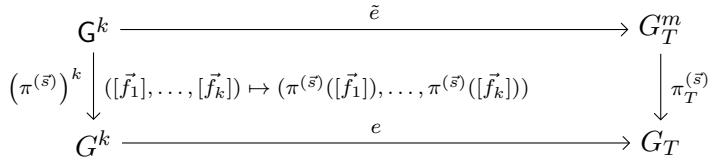

Figure 5.2: The projecting property

Lemma 5.6.2. *For any $\vec{s} \in \mathbb{Z}_p^d$, there exists some $\mu^{(\vec{s})} \in \mathbb{Z}_p^*$ such that $\pi^{(\vec{s})}(\vec{f}) = \mu^{(\vec{s})} \sum_{i=0}^{\ell} f_i \mathfrak{q}_i(\vec{s})$.*

Proof. Since $\mathbb{H}^{(\vec{s})}$ has co-dimension 1 in $G^{\ell+1}$, any two maps $G^{\ell+1} \to G$, both with kernels $\mathbb{H}^{(\vec{s})}$, differ by a non-zero scalar multiple. By definition, $\ker \pi^{(\vec{s})} = \mathbb{H}^{(\vec{s})}$. Since the map $\tilde{\pi}\colon\ G^{\ell+1} \to G$ which sends $\vec{f} \in G^{\ell+1}$ to $f(\vec{s}) = \sum_{i=0}^{\ell} f_i \mathfrak{q}_i(\vec{s})$ is another linear map with kernel $\mathbb{H}^{(\vec{s})} = \{\vec{f} \in G^{\ell+1} \mid f(\vec{s}) = 0\}$, the claim follows. □

Without loss of generality we assume in the following that $\mu^{(\vec{s})} = 1$ for all $\vec{s}$. This follows from the fact that if $\tilde{e}$ satisfies the projecting property with respect to the maps $\pi^{(\vec{s})}, \pi_T^{(\vec{s})}$ then the same property is satisfied by the maps $((\mu^{(\vec{s})})^{-1}\pi^{(\vec{s})})$ and $((\mu^{(\vec{s})})^{-k}\pi_T^{(\vec{s})})$.

For the second step, we consider the commutative diagram Figure 5.3 for an interpolating set $\vec{s}_1, \ldots, \vec{s}_{\overline{m}}$ for W.

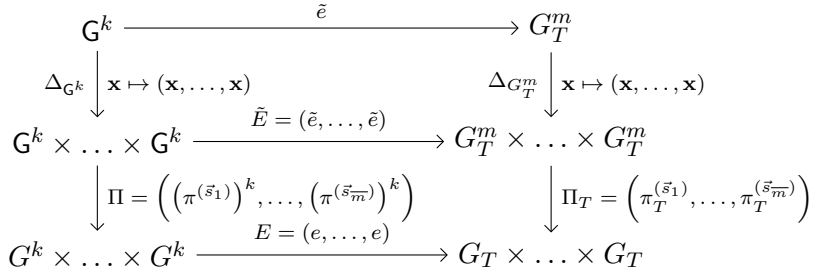

Figure 5.3: Figure 5.2 repeated $\overline{m}$ times for an interpolating set $\vec{s}_1, \ldots \vec{s}_{\overline{m}}$ for W.

With the above Lemma 5.6.2 it holds that

$$e\Big(\left(\pi^{(\vec{s}_i)}\right)^k([\vec{f_1}],\ldots,[\vec{f_k}])\Big) = e\Big([f_1(\vec{s}_i)],\ldots,[f_k(\vec{s}_i)]\Big) = [(f_1 \cdots f_k)(\vec{s}_i)]_T,$$

where $f_j(\vec{X})$ is the polynomial defined by $f_j(\vec{X}) = \sum_t \mathfrak{q}_t(\vec{X}) f_{j,t}$. It follows that, going first down, then right in the diagram,

$$E(\Pi(\Delta_{\mathsf{G}^k}([\vec{f_1}],\ldots,[\vec{f_k}]))) = ([(f_1 \cdots f_k)(\vec{s}_1)]_T,\ldots,[(f_1 \cdots f_k)(\vec{s}_{\overline{m}})]_T),$$

from which $f_1 \cdots f_k \in W$ can be interpolated via a linear map. It follows that the span of the image of $E \circ \Pi \circ \Delta_{\mathsf{G}^k}$ has dimension at least $\overline{m} = \dim W$. But traversing the diagram first right, then down, we see that the image of $E \circ \Pi \circ \Delta_{\mathsf{G}^k}$ is contained in $(\Pi_T \circ \Delta_{G_T^m})(G_T^m)$, where $\Pi_T \circ \Delta_{G_T^m}$ is a linear map. So the dimension of the span of the image of $E \circ \Pi \circ \Delta_{\mathsf{G}^k}$ can be at most $\dim G_T^m = m$. This implies $\overline{m} \leq m$, finishing the proof.

□

5.6.2 Optimality of our Projecting Multilinear Map from the SCasc-Assumption

As a result of our general viewpoint, we can actually show that the projecting multilinear map based on the SCasc-assumption is optimal among *all* polynomially induced matrix assumptions $\mathcal{D}_{n,\ell}$ that are not redundant. Non-redundancy rules out the case where some components of $\vec{z}$ are of no help (even information-theoretically) in distinguishing $\vec{z} \in \mathsf{G}$ from $\vec{z} \in \mathbb{H}^{(s)}$. See 5.5.1 for a formal definition of redundancy.

Theorem 5.6.3. *Let $n = \ell + 1$ and $\mathcal{D}_{n,\ell}$ be a polynomially induced matrix distribution which is not redundant. Let $(\mathcal{MG}_k, \mathbb{H}, G^n, G_T^m, \tilde{e})$ be the output of some projecting $(k, \mathcal{D}_{n,\ell})$ multilinear map generator with a fixed multilinear map. Then, $m \geq \ell k + 1$.*

Note that the projecting pairing based on the polynomial viewpoint of the ℓ-SCasc-assumption reaches this bound and is hence optimal.

Proof. We may identify G^n with some subspace $V \subset \mathbb{Z}_p[\vec{X}]$ of dimension n. By Theorem 5.6.1 above, we may assume w.l.o.g. that $\tilde{e}$ is polynomial multiplication, as this only makes m smaller. Hence we can also identify G_T^m with some subspace $W \subset \mathbb{Z}_p[\vec{X}]$ of dimension m. Let $>$ be any monomial ordering on $\mathbb{Z}_p[\vec{X}]$. Let $\mathfrak{q}_0, \ldots, \mathfrak{q}_\ell$ be a basis of V in echelon form with respect to $>$. This implies that the leading monomials satisfy $\mathrm{LM}(\mathfrak{q}_0) > \ldots > \mathrm{LM}(\mathfrak{q}_\ell)$. Now consider the elements

$$
\begin{array}{llll}
\mathfrak{q}_0^k = \mathfrak{r}_0 = \mathfrak{q}_0 \cdots \mathfrak{q}_0 \mathfrak{q}_0 & & & \\
\quad \mathfrak{r}_1 = \mathfrak{q}_0 \cdots \mathfrak{q}_1 \mathfrak{q}_0 & \mathfrak{r}_{\ell+1} = \mathfrak{q}_0 \cdots \mathfrak{q}_0 \mathfrak{q}_1 \mathfrak{q}_\ell & \cdots & \mathfrak{r}_{(k-1)\ell+1} = \mathfrak{q}_0 \mathfrak{q}_\ell \cdots \mathfrak{q}_\ell \\
\quad \vdots & \vdots & & \vdots \\
\quad \mathfrak{r}_\ell = \mathfrak{q}_0 \cdots \mathfrak{q}_0 \mathfrak{q}_\ell & \mathfrak{r}_{2\ell} = \mathfrak{q}_0 \cdots \mathfrak{q}_0 \mathfrak{q}_\ell \mathfrak{q}_\ell & \cdots & \mathfrak{r}_{\ell k} = \mathfrak{q}_\ell \mathfrak{q}_\ell \cdots \mathfrak{q}_\ell
\end{array}
$$

(the definition of $\mathfrak{r}_{i+1}$ differs from that of $\mathfrak{r}_i$ in one single index being greater by one). All $\mathfrak{r}_i \in W$ by construction and $\mathrm{LM}(\mathfrak{r}_0) > \mathrm{LM}(\mathfrak{r}_1) > \ldots > \mathrm{LM}(\mathfrak{r}_{\ell k})$ by the properties of a monomial order. It follows that the $\mathfrak{r}_i$ are linearly independent, showing $m = \dim W \geq \ell k + 1$. □

5.6.3 Extended Impossibility Results for Projecting and Canceling

In his original paper [45], Freeman gives several constructions of bilinear pairings which are either projecting or canceling — but not both. Subsequently, Meiklejohn *et al.* [89] give evidence that it might be hard to obtain both features simultaneously: they proved that there is no fixed projecting and canceling pairing for the $\mathcal{U}_\ell$ assumption. We restate their result here.

Theorem 5.6.4. *([89]) Any symmetric $(2, \mathcal{U}_\ell)$ bilinear generator with a fixed pairing cannot be simultaneously projecting and canceling, except with negligible probability (over the output of the generator).*[7]

[7] Their claim is that it is impossible to achieve both properties under what they call a "natural use" of the ℓ-Lin assumption although they are really using the uniform assumption.

It could be the case that, as it happened for the lower bounds for the image size, a change of assumption could suffice to construct a projecting and canceling pairing. However, the proof of [89] seems hard to generalize to other $\mathcal{D}_{n=\ell+1,\ell}$ assumptions. In this section, we give a very simple but limited extension of Freeman's result to any $(2, \mathcal{L}_\ell)$ and any $(2, \mathcal{SC}_2)$ bilinear generator. It remains an open question if the impossibility results extend to $(2, \mathcal{SC}_\ell)$, for $\ell > 2$.

Lemma 5.6.5. *Let $(k = 2, \mathbb{H}_1, G^n, \mathsf{G}_T, \tilde{e})$ be the output of a symmetric canceling $(k = 2, (n = \ell + 1, \ell))$ bilinear map generator. Then $\tilde{e}(\mathsf{G}^k) \subset \mathsf{G}_T$ is a vector space of dimension at most $\ell(\ell + 1)/2$.*

Proof. The map $\tilde{e}$ can be alternatively defined as a linear map from $\mathsf{G}\otimes\mathsf{G} \to \mathsf{G}_T$. First we note that, since $\tilde{e}$ is symmetric, the maximum dimension of the image of $\tilde{e}$ (which w.l.o.g. is G_T^m, for some $m \in \mathbb{N}$) is $(\ell + 1)(\ell + 2)/2$. This follows because the kernel of $\tilde{e}$ must contain all the symmetry relations , i.e., the span of all $\vec{e}_i\otimes\vec{e}_j - \vec{e}_j \otimes \vec{e}_i$. Additionally, since the map is canceling, and $\mathsf{G} = \mathbb{H}_1 \oplus \mathbb{H}_2$, it follows that $\mathbb{H}_1\otimes\mathbb{H}_2$ must also be in the kernel (note that if this is the case, by symmetry so is $\mathbb{H}_2 \otimes \mathbb{H}_1$). Since $\mathbb{H}_1 \cap \mathbb{H}_2 = \{0\}$, we have that $\mathbb{H}_1\otimes\mathbb{H}_2$ intersects the span of the symmetry relations only trivially. Since the dimension of $\mathbb{H}_1\otimes\mathbb{H}_2$ is ℓ, it follows that the size of the image is at most $m := \frac{(\ell+1)(\ell+2)}{2} - \ell = \frac{\ell(\ell+1)}{2} + 1$. □

The lemma also means that there is no $(2, \mathcal{L}_\ell)$ bilinear generator with a fixed pairing which is both projecting and canceling, because according to Section 5.6.1 the image size would be at least $\frac{(\ell+1)(\ell+2)}{2}$, while Lemma 5.6.5 says the image is at most $\frac{\ell(\ell+1)}{2} + 1$.[8] Further, we can prove that there is no $(2, \mathcal{SC}_2)$ bilinear generator with a fixed pairing which is both projecting and canceling (more generally, this extends to any $\mathcal{D}_{3,2}$ matrix distribution), since the optimality results of Section 5.6.1 and Section 5.6.2 imply that the image size would be at least 5 while Lemma 5.6.5 says the image size would be at most 4. It remains an open question to see if other impossibility results for ℓ-SCasc can be proven for $\ell > 2$.

With these impossibility results, it is not surprising that all projecting and canceling constructions are for *non-fixed* pairings in the sense of Definition 5.3.4. Indeed, in [108] Cheon and Seo construct a pairing which is both projecting and canceling but not fixed since, implicitly, the group G depends on the hidden subgroup $\mathbb{H}$. In our language, the pairing of Seo and Cheon is a $(2, (r = \ell^2, n = \ell + 1, \ell))$ pairing, i.e., $\mathsf{G} \subset G^{\ell^2}$ of dimension $n = \ell + 1$. Recently, Lewko and Meiklejohn [86] simplified this construction, obtaining a $(2, (r = 2\ell, n = \ell + 1, \ell))$ bilinear map generator. Our $(2, (r = 2\ell, n = \ell + 1, \ell))$ pairing achieving both properties from Section 5.4.3 (and which generalizes to any $(k, (r = 2\ell, n = \ell + 1, \ell))$ with $\ell \geq k$) uses completely different techniques. A direct comparison of [108], [86] with our pairing is not straightforward, since in fact they use dual vector spaces techniques and their pairing is not really symmetric.

[8] We note that this last result about $(2, \mathcal{L}_\ell)$ bilinear generators is not proven in [89]. Although the authors talk about a natural use of the ℓ-Lin assumption, their results are for the uniform assumption.

5.7 Efficiency Considerations for our Constructions

In some instantiations of multilinear settings, computing the (basic) mapping e is significantly more expensive than computing the group operation or even an exponentiation. For instance, this is the case for all instantiations of bilinear maps over elliptic curves we currently know. In such settings it might be worthwhile to strive for trade offs between applications of e and less expensive operations. In Section 5.7.1 we consider such trade offs for our projecting map constructions while Section 5.7.2 deals with the projecting and canceling maps.

5.7.1 Efficiency of the Projecting Constructions

For our constructions of projecting maps in Section 5.4.1 and Section 5.4.2, computing $\tilde{e}$ corresponds to usual multiplication of polynomials. Hence, we may apply methods for fast polynomial multiplication to reduce the number of applications of e. Concretely, we may follow an evaluate-multiply-interpolate approach. Consider the case that we are given the polynomials $f_1, \ldots, f_k$ all from a subspace V of dimension n (e.g., univariate polynomials of degree at most $n-1$), and we know that their product lies in a subspace W of dimension m (e.g., $k = 2$ and W contains all univariate polynomials of degree at most $2n-2$, so $m = 2n-1$). Then we can first evaluate all f_j at m publicly known points $x_0, \ldots, x_{m-1}$ that form an interpolating set for W, then multiply $f_1(x_i) \cdots f_k(x_i)$ for any i to obtain $(f_1 \cdots f_k)(x_i)$, from which our desired result $f_1 \cdots f_k$ can be interpolated. One thing to note here is that the map sending a polynomial $f \in W$ to the vector $f(x_0), \ldots, f(x_{m-1})$ is a bijective linear map, whose coefficients depend only on the publicly known x_i, so both evaluation and interpolation can be computed without any pairings. As a consequence, using this approach for computing $\tilde{e}$, we can reduce the number applications of e to m at the cost of having to apply some linear maps, which correspond to multi-exponentiations in G.

Intermediate trade offs are possible here. For instance, it is easy to see that the Karatsuba algorithm [78] can immediately be applied to our bilinear map based on 2-SCasc. This would reduce the number of basic pairing applications from a naive 9 to 6 (rather than all the way to 5) at the cost of only 9 additional group operations (e.g., see [116]). Note that there are also generalizations of Karatsuba to the multivariate case, e.g., [117].

Since interpolation is a publicly known linear map, this just corresponds to choosing a basis for W and has no effect on the hardness of subgroup indistinguishability. In particular, this means we do not need to interpolate in the end, but can simply use $[(f_1 \cdots f_k)(x_0)]_T, \ldots, [(f_1 \cdots f_k)(x_{m-1})]_T \in G_T^m$ as the final result, representing polynomials in the target space by their evaluations at the interpolating set.

This observation means that we should choose the interpolation points in such a way that computations of the map $\mathsf{G} \to G^m, [f] \mapsto ([f(x_0)], \ldots, [f(x_{m-1}])$ should be cheap. If vectors from G correspond to polynomials via coefficient representation, we can simply choose small x_i (usually, this has the downside of making the coefficients for interpolation large, which does not matter here). Concretely, for our 2-SCasc based construction, we chose interpolation points as $M = \{-2, -1, 0, 1, 2\}$. Given coefficients $[f_0], [f_1], [f_2]$ of a polynomial $f = f_0 + f_1 X + f_2 X^2$ of degree at most 2, one can compute $[f(-2)], [f(-1)], [f(0)], [f(1)], [f(2)]$ with only 11 addi-

tions/inversions. Furthermore, it is possible to amortize the cost of evaluation, if the same $[f]$ is used in several applications of $\tilde{e}$.

Computing $\pi\colon \mathsf{G} \to G$ for known $\vec{s}$ is a linear map and hence corresponds to one n-multi-exponentiation. For our SCasc-based constructions, this means computing $[f_0 + sf_1 + s^2 f_2 + \ldots]$ from $[f]$ and s. Since f_0 is not multiplied by anything, we really only have a $n-1$-multi-exponentiation and one group operation. For the computation of $\pi_T\colon \mathsf{G}_T \to G_T$, this latter saving is no longer possible if we represent elements from W by their evaluations. Instead, $[f(s)]_T = \pi_T([\mathcal{P}_0]_T, \ldots, [\mathcal{P}_{m-1}]_T)$ is computed as $\pi_T([\mathcal{P}_0]_T, \ldots, [\mathcal{P}_{m-1}]_T) = \sum_i \mathfrak{r}_i(\vec{s}) \cdot [\mathcal{P}_i]_T$, where the coordinate $[\mathcal{P}_i]_T$ corresponds to the value $\mathcal{P}_i = f(x_i)$ of some polynomial f at x_i. This corresponds to the basis $\mathfrak{r}_0, \ldots, \mathfrak{r}_{m-1}$ of W, determined by $\mathfrak{r}_i(x_j) = 0$ if $i \neq j$ and 1 otherwise. Note that the $\mathfrak{r}_i$ are known and computing $\mathfrak{r}_i(\vec{s})$ is just a computation in $\mathbb{Z}_p$ (which is fast).

5.7.2 Efficiency of the Projecting and Canceling Constructions

Let us briefly consider our projecting and canceling constructions from Section 5.4.3 based on variants of SCasc. Computation of π and π_T can be done as in the projecting construction. So let us turn our attention to the efficiency of $\tilde{e}$ respectively $\tilde{e}_{\text{ext}}$ with respect to the application of fast multiplication algorithms. Here the the situation is more intricate as we also need to perform modular reduction in the exponent. Furthermore, we chose $h'(X) = X^n$, which gives us an advantage if we stay in the coefficient representation, as the reduction modulo $X^n - s^n$ has an easier form then.

The naive way of computing either $\tilde{e}$ or $\tilde{e}_{\text{ext}}$ requires exactly n^k applications of the k'-linear e and $n^k - n^{k-1}$ additions in G_T. For $\tilde{e}_{\text{ext}}$ from the modified construction in Section 5.4.3, this is the best method we are aware of, both in the $\overline{\mathsf{G}_{\text{ext}}}$ and in the G_{ext} variant.

For $\tilde{e}$ from the first construction in Section 5.4.3, we can use some ideas from efficient polynomial multiplication to improve this. Perhaps the most simple idea which, however, only works in certain settings is the following: Let us first assume that we are given a k'-linear basic map e to implement our k-linear map $\tilde{e}$ as in the first construction in Section 5.4.3. Moreover, assume that e is not given as a "monolithic block" but as a series of pairings $e_{i,j} : G^{(i)} \times G^{(j)} \to G^{(i+j)}$ like it is the case for the currently known multilinear map candidates. In such a setting, it is possible to first compute products consisting of only k factors and then multiply (linear combinations of) these sub products with another factor. This enables us to first compute the coefficients of $[(f_1 \cdots f_k)]$ in $G^{(k)}$ using the fast polynomial multiplication algorithms as described before and subsequently, perform the modular reduction by multiplying these coefficients with the appropriate reduction term $[s^{in}]$ for appropriate i by means of $e_{k,1}$. Note that we can perform polynomial interpolation onto intermediate results, which means we can use a multiplication tree, reducing the number of interpolation points required for intermediate products. Also, we interpolate in the end, so the final modular reduction can be performed in the coefficient representation. This way, (only counting applications of e), for the multiplication, we need at most (or exactly if k is a power of 2) $k(n-1)\lceil \log_2 k \rceil + k - 1$ applications of some $e_{i,j}$, and for the reduction we need $k(n-1) - n$ applications of $e_{k,1}$. This makes a total of (at most) $k(n-1)\lceil 1 + \log_2 k \rceil + k - n - 1$ (bilinear) pairings. Note that this counts bilinear pairings, i.e., only "partial" k'-linear pairings and hence can

not be directly compared to applications of a k'-linear map.

Now, assume we are given a $k+1$-linear basic map as a black-box, i.e., not as a series of pairings. We use the evaluate-multiply approach as before, so consider the interpolating set $x_0, \ldots, x_{k(n-1)}$ with interpolation polynomials $\mathfrak{r}_i$ such that $\mathfrak{r}_i(x_j) = 0$ for $i \neq j$ and 1 otherwise. Let

$$\mathfrak{r}_i \bmod h = \sum_{j=0}^{n-1} \overline{h}_{i,j} X^j.$$

Note that the $[\overline{h}_{i,j}]$'s are computable from the $[h_{i,j}]$'s and x_i's. Then we can compute $\tilde{e}([f_1], \ldots, [f_k])$ as

$$\tilde{e}([f_1], \ldots, [f_k]) = [f_1 \cdots f_k \bmod h]_T = \left[\sum_{j=0}^{n-1} \sum_{i=0}^{k(n-1)} \overline{h}_{i,j} f_1(x_i) \cdots f_k(x_i) X^j \right]_T.$$

This requires $kn(n-1)+n$ applications of e. Note that we can not make use of the special form of $h(X) = X^n - s^n$ this way and this is worse than the naive approach for small values of n, k (but much better asymptotically). Also for small values of n and k and h of a general form, there are dedicated tricks to reduce the number of basic map applications. For instance, in the case $k=2$, $n=3$, and general h, we may compute $\tilde{e}_{\text{ext}}$ (which is defined in a similar way as our modified construction in Section 5.4.3 for special $h = X^n - s^n$) using 12 applications of e compared to 15 using the naive approach.

5.8 Applications

In this section, we will exemplarily illustrate how applications benefit from our more efficient and general constructions. I.e., using our projecting pairing from Section 5.4.1, we can improve the performance of Groth-Sahai proofs by almost halving the number of required prime-order pairing operations (cf. Table 5.1). Additionally, in Section 5.8.2, we show how to implement a k-linear variant of the Boneh-Goh-Nissim encryption scheme [13] using the projecting multilinear map generator $\mathcal{G}_{k,\mathcal{SC}_k}$.

5.8.1 Instantiating Groth Sahai Proofs

BASICS ABOUT GROTH SAHAI PROOFS. Groth Sahai proofs are NIZK proofs of satisfiability of a set of equations in a bilinear group $\mathcal{MG}_2 := (2, G, G_T, e, p, \mathcal{P}, \mathcal{P}_T)$. The proofs follow a basic commit-and-prove approach (for a formalization of this see [41]) in which the witness for satisfiability (some elements in G or in $\mathbb{Z}_p$, depending on the equation type) is first committed to and then the proof shows that the committed value satisfies the equation. The common reference string includes some commitment keys which can be generated in two computationally indistinguishable ways: in the soundness setting, the keys are perfectly binding and in the witness indistinguishability setting they are perfectly hiding.

Groth-Sahai proofs [63] are the most natural application of projecting bilinear maps. They admit various instantiations in the prime-order setting. It follows easily

from the original formulation of Groth and Sahai that their proofs can be instantiated based on any $\mathcal{D}_{n,\ell}$ assumption and any fixed projecting map. Details are given in [42] but only for the projecting pairing corresponding to the symmetric bilinear tensor product. The only point where our construction differs from the one given in ([42], section 4.4) is in the definition of the symmetric bilinear map F, which we define to be $\tilde{e}$, the projecting bilinear map corresponding to polynomial multiplication defined in Section 5.4.2. The pairing $\tilde{e}$ is described by a tuple $(\mathcal{MG}_2, [\mathbf{A}], \mathsf{G} = G^{\ell+1}, G_T^m, \tilde{e})$, $\mathbf{A} \leftarrow \mathcal{D}_\ell$. The only information related to F which has to be included in the common reference string are some matrices $[\mathbf{H}_1], \ldots, [\mathbf{H}_\eta]$ whose purpose is to ensure that the proof is correctly distributed among all proofs satisfying the verification equation.

Define for any two vectors of elements of G of equal length r, $[\vec{X}] = ([\vec{x}_1], \ldots, [\vec{x}_r])$, $[\vec{Y}] = ([\vec{y}_1], \ldots, [\vec{y}_r])$, the maps $\bullet$ associated with $F = \tilde{e}$ as $[\vec{X}] \bullet [\vec{Y}] = \sum_{i=1}^r \tilde{e}([\vec{x}_i], [\vec{y}_i])$. More specifically, the information which depends on F which is in the setup is the following (depending on the equation type):

- **Pairing product equations**. In this case, $\mathbf{H}_1, \ldots, \mathbf{H}_\eta$ are a basis of the space of all matrices which are a solution of the equation $[\mathbf{UH}] \bullet [\mathbf{U}] = [\mathbf{0}]_T$, where $\mathbf{U}$ is the commitment key. (This commitment key is either of the form $[\mathbf{U}] = [\mathbf{A}||\mathbf{A}\vec{w}]$ or $[\mathbf{U}] = [\mathbf{A}||\mathbf{A}\vec{w} - \vec{z}]$ for random $\vec{w}$ and a public $\vec{z} \notin \mathrm{Im}(\mathbf{A})$.)
- **Multi-scalar multiplication equations**. In this case, $\mathbf{H}_1, \ldots, \mathbf{H}_\eta$ are a basis of the space of all matrices which are a solution of the equation $[\mathbf{AH}] \bullet [\mathbf{U}] = [\mathbf{0}]_T$.
- **Quadratic equations**. In this case, $\mathbf{H}_1, \ldots, \mathbf{H}_\eta$ are a basis of the space of all matrices which are a solution of the equation $[\mathbf{AH}] \bullet [\mathbf{A}] = [\mathbf{0}]_T$.

We discuss how these matrices ought to be defined when $F = \tilde{e}$, polynomial multiplication via the identification between polynomials and vectors in $G^{\ell+1}$ defined by $\mathcal{D}_\ell$. The matrices are independent of the choice of basis for the image space W, since a change of basis corresponds to multiplication by an invertible matrix. Therefore, these matrices can be chosen depending only on $\mathcal{D}_\ell$, without having to specify W.

For the pairing of Seo [107] and which corresponds also to our construction for $\mathcal{U}_\ell$ and ℓ-Lin, these matrices are the same as the ones given in [42], namely matrices of the appropriate size which encode the symmetric relations which are in the kernel of $\tilde{e}$. On the other hand, for ℓ-SCasc, additional relations — apart from the ones derived from symmetry — appear only for pairing product equations. For concreteness, we give an exact description of the matrices for the 2-SCasc assumption:

- **Pairing product equations**. A choice of basis is:

$$\mathbf{H}_1 := \begin{pmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \mathbf{H}_2 := \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{pmatrix}, \mathbf{H}_3 := \begin{pmatrix} s & 0 & 0 \\ -2s & s & 0 \\ 2s & 1-2s & s-1 \end{pmatrix},$$

  where $s \in \mathbb{Z}_p$ describes $\mathbf{A}$, namely $\mathbf{A} = \mathbf{A}(s)$.

- **Multi-scalar multiplication equations**. A choice of basis is:

$$\mathbf{H}_1 := \begin{pmatrix} 0 & 1 \\ -1 & 0 \\ 0 & 0 \end{pmatrix}.$$

- **Quadratic equations**. A choice of basis is:

$$\mathbf{H}_1 := \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}.$$

It can be easily seen that compared to the assumptions $\mathcal{U}_2$, and 2-Lin (see [63, 42]), the only difference is the matrix $\mathbf{H}_3$. As we announced, the intuition is that $\mathbf{H}_i$, $i \neq 3$, encode the symmetric relations in the kernel of $\tilde{e}$, while for PPE's an additional element in the kernel appears (which accounts for the smaller image size of our pairing).

EFFICIENCY DISCUSSION. The important efficiency measures for GS proofs are common reference string size, proof size, prover's and verification's efficiency. The proof size (for a given equation) depends only on the size of the matrix assumption, that is of n, ℓ, so it is omitted in our comparison. Essentially, so does the efficiency of the prover, with some minor differences which are discussed below. For the rest, the discussion goes as follows:

- **Size of the common reference string.** ℓ-SCasc assumption is the most advantageous from this point of view (as noted in [42]) since the commitment keys can be described more compactly in this case. Note that the matrices $[\mathbf{H}_i]$ do not have to be published since they are fixed and computable from the common reference string. The size of the common reference string depends essentially one size of the commitment key, which is $n + \mathrm{Re}_{\mathsf{G}}(\mathcal{D}_{n,\ell})$, where $\mathrm{Re}_{\mathsf{G}}(\mathcal{D}_{n,\ell})$ is the representation size of the matrix assumption $\mathcal{D}_{n,\ell}$, which is 1 for ℓ-SCasc, ℓ for ℓ-Lin and $(\ell+1)\ell$ for $\mathcal{U}_\ell$. Therefore, the ℓ-SCasc instantiation is the most advantageous from the point of view of the size of the common reference string (regardless of the pairing used), as pointed out in [42].

- **Prover's computation.** The cost of computing a commitment depends roughly on the sparseness of the matrix $\mathbf{A}$. For instance, for the uniform assumption with $\ell = 2$, a commitment costs at least 6 exponentiations, so $\mathrm{lin}2$ and 2-SCasc are more advantageous. On the other hand, the instantiation of proofs for PPE using 2-SCasc requires in comparison to compute additionally $r\mathbf{H}_3$, for some $r \leftarrow \mathbb{Z}_p$. This can be done very efficiently by computing simply $[rs]$ and $[r]$ and obtaining $r\mathbf{H}_3$ via doubling and the group operation. Following [41], the prover's computation can be reduced significantly by allowing a prover to choose its own common reference string and then proving its correctness. This allows the prover to minimize the number of exponentiations, since the prover knows $s \in \mathbb{Z}_p$ and can compute most operations in $\mathbb{Z}_p$. Obviously the same trick applies here.

- **Verification cost** Verification cost is the efficiency measure which depends more on the choice of pairing, as it typically involves several evaluations of the map F. Since the map $\tilde{e}$ can be computed more efficiently than F, the verification cost is significantly reduced for many equation types. For instance, using our map derived from 2-SCasc we can save 4 basic pairing evaluations per evaluation of $\tilde{e}$.

We emphasize that this discussion is for general equation types. For some specific types — like linear equations with constants in G, the new map does not imply more efficient verification.

We conclude that the 2-SCasc instantiation with polynomial multiplication is definitely the most efficient implementation for GS NIZK proofs in symmetric bilinear map, not only because of the size of the common reference string as pointed out in [42] but also from point of view of efficiency of verification. For instance, this leads to a saving of 12 pairing evaluations for proving that a committed value is a bit $b \in \{0, 1\}$.

5.8.2 Efficient Implementation of the k-times Homomorphic BGN Cryptosystem

In this section we show how to implement a multilinear variant of the Boneh-Goh-Nissim (BGN) encryption scheme from [13] with prime-order multilinear groups. We proceed as follows: first we transform a given prime-order multilinear group into a projecting composite-order multilinear group using the results from Section 5.4.2. As seen in Section 5.6.2, the most efficient way to do this is using the k-SCasc assumption. We write the generator, already given in Example 5.4.2, again in more detail. In the next step, we show how to implement the BGN cryptosystem in those groups and compare the implementation costs to implementations of k-BGN derived from the work of Freeman ([45]) and Seo ([107]).

Example 5.8.1 (A generator for the BGN cryptosystem)**.** *Let $k \in \mathbb{N}$ and $\mathcal{SC}_k$ denote the matrix distribution belonging to the symmetric cascade assumption from Definition 2.4.4. Let $\mathcal{G}_{k-BGN}$ be an algorithm that, on input a security parameter $\lambda \in \mathbb{N}$, does the following:*

- *Obtain $\mathcal{MG}_k := (k, G, G_T, e, p, \mathcal{P}, \mathcal{P}_T)$ from a symmetric k-linear group generator $\mathcal{G}_k(\lambda)$.*
- *Let $\mathsf{G} := G^{k+1}, \mathsf{G}_T := G_T^{k^2+1}$.*
- *Choose $s \xleftarrow{R} \mathbb{Z}_p$ and let $\mathbb{H}^{(s)} \subset \mathsf{G}, \mathbb{H}_T^{(s)} \subset \mathsf{G}_T$ as in Section 5.4.1.*
- *Let $\tilde{e}([\vec{f}_1], \ldots, [\vec{f}_k]) := [f_1 \cdots f_k]_T$.*
- *Output the tuple $(\mathcal{MG}_k, \mathbb{H}^{(s)}, \mathsf{G}, \mathsf{G}_T, \tilde{e})$.*

Observe that $\mathcal{G}_{k-BGN}$ is a $(k, \mathcal{SC}_k)$ multilinear map generator. Additionally, $\mathcal{G}_{k-BGN}$ is projecting for $\mathbb{H}^{(s)}$ w.r.t. the maps $\pi\colon \mathsf{G} \to G$, $[\vec{f}] \mapsto [f(s)]$ and $\pi'\colon \mathsf{G}_T \to G_T$, $[\vec{f}]_T \mapsto [f(s)]_T$. Both maps can be computed efficiently given s. From the discussion in Section 5.4.1 it follows that if $\mathcal{G}_k$ satisfies the k-SCasc assumption, then $\mathcal{G}_{k-BGN}$ satisfies the subgroup indistinguishability property. As seen in Section 5.4.2, the computation of the map $\tilde{e}$ can be optimized using techniques for fast polynomial multiplication.

The BGN scheme as introduced in [13] is additively and one time multiplicatively homomorphic. It uses a pairing and can be extended in a straightforward way to work with a k-linear map for arbitrary $k \in \mathbb{N}$. The resulting encryption scheme is then $k-1$ times multiplicatively homomorphic. We now describe how to implement the scheme using $\mathcal{G}_{k-BGN}$.

Setup(1^λ): Run $\mathcal{G}_{k-BGN}$ to obtain $(\mathcal{MG}_k, \mathbb{H}^{(s)}, \mathsf{G}, \mathsf{G}_T, \tilde{e})$. Afterwards, output $\mathcal{PK} := (p, \mathsf{G}, \mathsf{G}_T, \tilde{e}, \mathcal{P}, [s])$ and $\mathcal{SK} := s$.

Enc($\mathcal{PK}, m$): To encrypt a message in G, draw $h_0, \ldots, h_k \xleftarrow{R} \mathbb{Z}_p \setminus \{0\}$ and compute $h := [(-sh_0, h_0 - sh_1, \ldots, h_{k-1} - sh_k, h_k)] \in \mathbb{H}^{(s)}$ using $[s]$ from $\mathcal{PK}$. Set $\mathcal{P}^0 := [(1, 0, \ldots, 0)]$. Compute and output the ciphertext as

$$c := m \cdot g^0 + h = [(-sh_0 + m, h_0 - sh_1, \ldots, h_{k-1} - sh_k, h_k)].$$

Encryption in G_T works similarly.

Dec($\mathcal{SK}, c$): Decryption in G and G_T works by applying π, i.e., evaluating c, interpreted as polynomial $c(X)$, in $\mathcal{SK} = s$. For this, parse $c := (c_1, \ldots, c_l)$ and compute $\pi(c) = [c(s)] = [c_1 + s \cdot c_2 + \cdots + s^l \cdot c_l]$. Output $m = \log_g(c(s))$.

Add(PK, c, c'): We assume $c, c' \in \mathsf{G}$. Draw $\hat{h} \xleftarrow{R} \mathbb{H}^{(s)}$. Compute and output

$$c + c' + \hat{h} = (m + m') \cdot g^0 + h + h' + \hat{h}.$$

Adding encrypted messages in G_T works just as in G.

Mult($\mathcal{PK}, c_1, \ldots, c_k$): We require $c_1, \ldots, c_k \in \mathsf{G}$. Draw $\hat{h} \xleftarrow{R} \mathbb{H}_T^{(s)}$. Compute and output

$$\tilde{e}(c_1, \ldots, c_k) + \hat{h} = (m_1 \cdot \ldots \cdot m_k) g_T^0 + \tilde{h} + \hat{h},$$

where $\mathcal{P}_T^0 := [(1, 0, \ldots, 0)]_T$ and $\tilde{h} \in \mathbb{H}^{(s)}$.

Observe that correctness of decryption follows from $c(s) = [-sh_0 + m + s(h_0 - sh_1) + \cdots + s^{k-1}(h_{k-1} - sh_k) + s^k(h_k)] = [m + h(s)] = [m]$. The number of G-exponentiations required for encryption and decryption is equal to the number of copies of G used for G and G_T, i.e., $k + 1$ for G and $k^2 + 1$ for G_T.

Corollary 5.8.2. *The above scheme is semantically secure if the group generator $\mathcal{G}_k$ satisfies the k-*SCasc *assumption.*

Proof. Semantic security follows from a straightforward adaption of Theorem 3.1 from [13] and the fact that $\mathcal{G}_{k-BGN}$ satisfies the subgroup indistinguishability property if $\mathcal{G}_k$ satisfies the k-SCasc assumption. □

COMPARISON TO AN EXTENSION OF FREEMAN'S CONSTRUCTION ([45]). The projecting pairing from [45] has a natural extension to the multilinear case. For $k \in \mathbb{N}$, the k-linear extension of the symmetric bilinear generator of [45], Theorem 2.5, is a $(k, \mathcal{U}_k)$ multilinear map generator (note that Freeman uses the uniform distribution to generate subgroups). We can define the k-linear map such that it is projecting, following [45], Section 3.1 (using the notation from the original paper). Thus, we let the generator compute $\tilde{e}$ as $\tilde{e}([\vec{f}_1], \ldots, [\vec{f}_k]) := e(g, \ldots, g)^{\vec{f}_1 \otimes \cdots \otimes \vec{f}_k}$. This setting can be further optimized for multilinear maps if we use an asymmetric prime-order map as a starting point for an asymmetric generator. We will not go into the details, since the construction is essentially the same as in [45], Example 3.3, naturally extended to the multilinear setting. On a high level, the main advantage is that we can keep the dimension of the subgroups and thus the composite-order groups small (i.e., $\mathsf{G}_i := G_i^2$), leading to a smaller (though still exponentially large) number of basic multilinear map evaluations to compute $\tilde{e}$. Note that even the asymmetric generator, using $\mathsf{G}_i = G_i^2$, requires $2k$ group elements to describe ciphertexts in the base groups. This is because the BGN cryptosystem has to be adjusted to work with an asymmetric map (see [45], Section 5, for details).

Generator	ciphertexts (in G/G_T)		Enc/Dec (in G/G_T)		Mult	
	(el. from G)	(el. from G_T)	(exp. in G)	(exp. in G_T)	(exp. in G_T)	(eval. of e)
Freeman, symm.	$k+1$	$(k+1)^k$	$(k+1)^2$	$(k+1)^{2k}$	-	$(k+1)^{k+1}$
Freeman, asymm.	$2k$	$2^k \cdot k$	2^2	2^{2k}	-	2^k
Seo, symmetric	$k+1$	$\binom{2k}{k}$	$(k+1)^2$	$\binom{2k}{k}^2$	-	$(k+1)^k$
Our $\mathcal{G}_{k-BGN}$	$k+1$	k^2+1	$k+1$	k^2+1	-	$(k+1)^k$
Our $\mathcal{G}_{k-BGN}$, opt.	$k+1$	k^2+1	$k+1$	k^2+1	$(k^3+k)k$	k^2+1

Table 5.2: Implementation costs and ciphertext sizes of k-BGN using different generators.

COMPARISON TO AN EXTENSION OF SEO'S CONSTRUCTION ([107]). As we explained in Section 5.3, in the constructions of Freeman, subgroups are always sampled according to the uniform assumption. Under this condition, Seo ([107]) proved that Freeman's construction of a projecting pairing is not optimal in the symmetric case. For this case, Seo gives a projecting pairing that is optimal in Freeman's model and, as seen in Section 5.9, matches our construction for the uniform assumption, and can therefore be generalized to the multilinear case (see Example 5.4.4).

The implementation costs and ciphertext sizes of k-BGN using different generators are described in Table 5.2. We used the generators obtained by extending the construction of Freeman, the generator $\mathcal{G}_{k,\mathcal{U}_k}$ used by Seo ([107]) and our most efficient generator $\mathcal{G}_{k-BGN}$. The latter is listed twice, differing in the method to compute the mapping $\tilde{e}$ (naively or optimized using techniques for fast polynomial multiplication.) Costs are stated in terms of application of the basic map e and exponentiations in G, respectively G_T. To keep the exposition simple, we measure ciphertext sizes of the coefficient representation (not the optimized point-value representation introduced in Section 5.4). Observe that our construction is the only one for which trade offs between basic multilinear map evaluations and exponentiations are known.

5.9 A Unified View on Different Projecting Pairings From the Literature

In this section, we compare our constructions for the special case of a 2-linear map with previous constructions of Groth and Sahai[9] ([63]) and Seo ([107]). We use the language of Seo to represent all constructions consistently. Let us first briefly introduce the required tools for this.

Given two vectors $\vec{x} = (x_0, \ldots, x_{n-1}) \in \mathbb{Z}_p^n$ and $\vec{y} = (y_0, \ldots, y_{n-1}) \in \mathbb{Z}_p^n$, the tensor product $\vec{x} \otimes \vec{y}$ is defined as $(x_0y_0, x_0y_1, x_0y_2, \ldots, x_{n-1}y_{n-1})$. Any bilinear map $\tilde{e}\colon G^n \times G^n \to G_T^m$ can be uniquely described by a matrix $\mathbf{B} \in \mathbb{Z}_p^{n^2 \times m}$ such that $\tilde{e}([\vec{x}], [\vec{y}]) = e([1], [1])^{(\vec{x}\otimes\vec{y})\ \mathbf{B}} = [(\vec{x} \otimes \vec{y})\ \mathbf{B}]_T$.

We can now present the pairing $\tilde{e}$ of each construction in terms of the matrix $\mathbf{B}$.

[9] The most efficient symmetric construction of Freeman ([45]), based on 2-Lin, matches the one of Groth and Sahai and is thus not listed here.

1. Symmetric tensor product (original Groth Sahai construction):

$$\mathbf{B} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1/2 & 0 & 1/2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/2 & 0 & 0 & 0 & 0 & 1/2 & 0 \\ 0 & 1/2 & 0 & 1/2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1/2 & 0 & 0 & 1/2 \\ 0 & 0 & 1/2 & 0 & 0 & 0 & 0 & 1/2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1/2 & 0 & 0 & 1/2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}.$$

2. Seo's construction:

$$\mathbf{B} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \in \mathbb{Z}_p^{9\times 6}.$$

   Seo's construction can be also written as

$$\tilde{e}([\vec{x}], [\vec{y}]) := [(x_0y_0, x_0y_1 + x_1y_0, x_0y_2 + x_2y_0, x_1y_1, x_1y_2 + x_2y_1, x_2y_2)]_T.$$

   Seo proves that his construction is projecting for the $\mathcal{U}_2$ Assumption. We note that our construction for 2-Lin and $\mathcal{U}_2$ is exactly the same if we choose as a basis for W the set $\{\mathfrak{q}_i\mathfrak{q}_j : 0 \leq i \leq j \leq 2\}$, for the polynomials $\mathfrak{q}$ defined in Example 5.4.3 and Example 5.4.4, respectively.

3. Our construction for the $\mathcal{SC}_2$ assumption, choosing $\{1, X, X^2, X^3, X^4\}$ as a basis for W:

$$\mathbf{B} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \in \mathbb{Z}_p^{9\times 5}.$$

   Our construction can be also be written as

$$\tilde{e}([\vec{x}], [\vec{y}]) := [(x_0y_0, x_0y_1 + x_1y_0, x_0y_2 + x_2y_0 + x_1y_1, x_1y_2 + x_2y_1, x_2y_2)]_T.$$

4. Our construction for the $\mathcal{SC}_2$ assumption with an alternative choice for the basis of W:

$$\mathbf{B} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \mathbf{T} \in \mathbb{Z}_p^{9\times 5}, \quad \text{where} \quad \mathbf{T} := \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ -2 & -1 & 0 & 1 & 2 \\ 4 & 1 & 0 & 1 & 4 \\ -8 & -1 & 0 & 1 & 8 \\ 16 & 1 & 0 & 1 & 16 \end{pmatrix}.$$

This construction can be also be written as

$$\tilde{e}([\vec{x}],[\vec{y}]) := [\sum_{t=0}^{4}\sum_{i+j=t} x_i y_j (-2)^t, \sum_{t=0}^{4}\sum_{i+j=t} x_i y_j (-1)^t, x_0 y_0, \sum_{t=0}^{4}\sum_{i+j=t} x_i y_j, \sum_{t=1}^{4}\sum_{i+j=t} x_i y_j 2^t]_T.$$

5.9.1 Efficiency Improvement for Seo's Construction

Seo claims that his pairing (item (2) above) is optimal among all based on the uniform subgroup decision assumption in terms of a) image size and b) number of basic pairing operations. We restate the original theorem here.

Theorem 5.9.1. *([107]) Let $\mathcal{G}_{2,\mathcal{U}_\ell}$ be any (symmetric) projecting $(2,\mathcal{U}_\ell)$ bilinear map generator with output $(\mathcal{MG}_2, \mathbb{H}, \mathsf{G}, G_T^m, \tilde{e})$. Then (a) we have $m \geq (\ell+1)(\ell+2)/2$, and (b) the map $\tilde{e}$ cannot be evaluated with less than $(\ell+1)^2$ prime-order pairing operations.*

Regarding a), our results do not contradict Seo's claim. Regarding b), Seo claims that the number of basic pairing operations is at least the weight of the matrix $\mathbf{B}$, which is 9 for the $\mathcal{U}_2$ Assumption.

Seo's implicit assumption behind this seems to be that if the pairing has the form $(\vec{x}\otimes\vec{y})\mathbf{B}$ for some matrix $\mathbf{B}$, then the best way to compute it is also via such a vector-matrix product: for each non-zero entry $B_{i,j}$ of B, compute $B_{i,j}x_iy_j$ and then perform some additions in the exponent. This then corresponds to one pairing operation (computing products of x_i and y_j in the exponent, $B_{i,j}$ is a scalar) per non-zero entry of $\mathbf{B}$. We reduce this to the rank of $\mathbf{B}$ by applying linear transformations to $\vec{x}$, $\vec{y}$ prior to multiplication (more precisely, treating $\vec{x}$ and $\vec{y}$ as polynomials and interpolating).

More formally, for any pairing $\tilde{e}$ with associated matrix $\mathbf{B}$, to reduce the number of basic pairing operations to m, it suffices to find matrices $\mathbf{C} \in \mathbb{Z}_p^{(\ell+1)\times m}, \mathbf{D} \in \mathbb{Z}_p^{m\times m}$ such that

$$\tilde{e}([\vec{x}],[\vec{y}]) := [(\vec{x}\otimes\vec{y})\ \mathbf{B}]_T = [((\vec{x}\mathbf{C})\otimes(\vec{y}\mathbf{C}))\begin{pmatrix}\mathbf{I}_m \\ \mathbf{0}_{(m^2-m)\times m}\end{pmatrix}\mathbf{D}]_T.$$

Given these matrices, the pairing $\tilde{e}$ can be computed with only m evaluations of e, regardless of the weight of $\mathbf{B}$, as follows:

1. Compute $[\vec{u}] = [\vec{x}\mathbf{C}] \in G^m$ and $[\vec{y}] := [\vec{y}\mathbf{C}] \in G^m$.
2. Compute $[\vec{w}]_T = (e([u_1],[v_1]),\ldots,e([u_m],[v_m])) = ([u_1v_1]_T,\ldots,[u_mv_m]_T) = [(\vec{u}\otimes\vec{v})\begin{pmatrix}\mathbf{I}_m\\ \mathbf{0}_{m^2-m}\end{pmatrix}]_T$.
3. Compute the final $[\vec{z}]_T$ as $[\vec{z}]_T = [\vec{w}\mathbf{D}]_T$.

Note that steps 1,3 require only group operations in G, G_T.

In the specific case of Seo's construction (item (2) in the list) — which matches our construction for $\mathcal{U}_2$ for the basis $\{\mathfrak{q}_i\mathfrak{q}_j \mid 0 \le i \le j \le 2\}$ of W —, $m = 6$ and the matrices $\mathbf{C} \in \mathbb{Z}_p^{3\times 6}, \mathbf{D} \in \mathbb{Z}_p^{6\times 6}$ are defined as:

$$\mathbf{C} := \begin{pmatrix}\mathfrak{q}_0(\vec{x}_1) & \ldots & \mathfrak{q}_0(\vec{x}_6)\\ \mathfrak{q}_1(\vec{x}_1) & \ldots & \mathfrak{q}_1(\vec{x}_6)\\ \mathfrak{q}_2(\vec{x}_1) & \ldots & \mathfrak{q}_2(\vec{x}_6)\end{pmatrix}, \qquad \mathbf{D} := \begin{pmatrix}(\mathfrak{q}_0\cdot\mathfrak{q}_0)(\vec{x}_1) & \ldots & (\mathfrak{q}_0\cdot\mathfrak{q}_0)(\vec{x}_6)\\ (\mathfrak{q}_0\cdot\mathfrak{q}_1)(\vec{x}_1) & \ldots & (\mathfrak{q}_0\cdot\mathfrak{q}_1)(\vec{x}_6)\\ (\mathfrak{q}_0\cdot\mathfrak{q}_2)(\vec{x}_1) & \ldots & (\mathfrak{q}_0\cdot\mathfrak{q}_2)(\vec{x}_6)\\ (\mathfrak{q}_1\cdot\mathfrak{q}_1)(\vec{x}_1) & \ldots & (\mathfrak{q}_1\cdot\mathfrak{q}_1)(\vec{x}_6)\\ (\mathfrak{q}_1\cdot\mathfrak{q}_2)(\vec{x}_1) & \ldots & (\mathfrak{q}_1\cdot\mathfrak{q}_2)(\vec{x}_6)\\ (\mathfrak{q}_2\cdot\mathfrak{q}_2)(\vec{x}_1) & \ldots & (\mathfrak{q}_2\cdot\mathfrak{q}_2)(\vec{x}_6)\end{pmatrix}^{-1},$$

where $\mathfrak{q}_0(\vec{X}) = X_{21}X_{32} - X_{22}X_{32}$, $\mathfrak{q}_1(\vec{X}) = X_{11}X_{32} - X_{12}X_{31}$, $\mathfrak{q}_2(\vec{X}) = X_{11}X_{22} - X_{12}X_{21}$ and $\vec{x_i} \in \mathbb{Z}_p^6$ are any interpolating set for the space spanned by $\{\mathfrak{q}_i\mathfrak{q}_j \mid 0 \le i \le j \le 2\}$, which guarantees that $\mathbf{D}$ is properly defined. This allows us to bring down the number of basic pairing operations to only 6 instead of 9, which was the number of operations which Seo claims to be necessary for compute $\tilde{e}$.

Note that by changing the choice of basis for W we can also get an even more efficient projecting pairing for the uniform assumption. In the language we just introduced, this amounts to choose $\mathbf{C}$ as above but define $\mathbf{D}$ as the identity matrix. This allows us to save all the exponentiations in G_T.

5.10 Implementation with Multilinear Map Candidates

In this section, we investigate to what extent our constructions can be implemented with the recent candidates [51, 3] of approximate multilinear maps or the graded encoding scheme from Chapter 3. We stress that the candidate construction from [51] is currently not believed to support any matrix assumption and thus can not be used to implement our results. Nonetheless, we discuss it here as a representative for any (future) constructions realizing the version of a GES as introduced in [51].

5.10.1 Using the Candidate Multilinear Maps from [51]

The work of Garg, Gentry and Halevi [51] only provides approximations of multilinear maps in the following sense. Namely, instead of group elements, [51] define "noisy encodings." Essentially, a noisy encoding is a group element with an additional noise term. This means that there is a whole set of encodings $\mathsf{Enc}(g)$ of a group element $\mathcal{P}$. Each operation on encodings increases the size of their noise terms. (More specifically, the noise term of the result of an operation is larger than the noise terms of the inputs.) In particular, each encoding can be used only for an

a-priori limited number of operations. After that, its noise term becomes too large, and errors in computations may occur.

This noisy encoding of group elements has a number of effects which are relevant for our constructions:

Group membership hard to decide. It is not efficiently decidable whether a given encoding actually encodes *any* group element (with a certain noise bound).

Non-trivial comparisons. To determine whether two given encodings encode the same group element (i.e., lie in the same set $\mathsf{Enc}(g)$), we require a special comparison algorithm (which however can be made publicly available).

Non-unique computations. Even if two computations yield encodings of the same group element, the actual encodings may differ. Specifically, an encoding may leak (through its noise term) the sequence of operations used to construct it. To hide the sequence of performed operations, there exists a re-randomization algorithm that re-randomizes the noise term (essentially by adding a substantially larger noise term).

Black-box exponents. It is possible to choose (almost) uniformly distributed exponents, but these can only be used in a black-box way (using addition, subtraction, and multiplication), and without using their explicit integer representation.

Subgroup membership problems. The construction in [51] allows for a very generic attack on subgroup membership assumptions in the (encoded) "source group" of the multilinear map. In particular, matrix assumptions like SCasc or the ℓ-linear assumption do not appear to hold in the source group.

We now inspect our constructions for compatibility with approximate multilinear maps as sketched above. Syntactically, our constructions (from Section 5.4.2 and Section 5.4.3) start from a given group G and a k-linear map $e\colon G^k \to G_T$, and construct another group $\mathsf{G} = G^n$, along with $\mathsf{G}_T = G_T^m$ and a k-linear map $\tilde{e}\colon \mathsf{G}^k \to \mathsf{G}_T$. In both cases, computations in G, G_T, and the evaluation of $\tilde{e}$ can be reduced to computations in G, G_T, and evaluating e. Hence, at least syntactically, our constructions can be implemented also with approximate multilinear maps as above. But of course, this does not mean that our constructions also retain the security properties we have proved when implemented in an approximate setting. Hence, we now investigate the effect of the imperfections sketched above.

Group membership hard to decide. We have assumed that group membership in our constructed group G is easy to decide. This of course no longer holds if the underlying prime-order group cannot be efficiently decided in the first place. We stress that this has no implications on *our* results, but of course makes the constructed group G also less useful in applications.

Non-trivial comparisons. Since, in our constructions, we never explicitly use comparisons, we also never need to use a comparison algorithm. On the other hand, a comparison in the groups G and G_T we construct can be reduced to comparing elements of G and G_T.

Non-unique computations. In the (encoded) groups we construct, the noise of the underlying G- or G_T-elements also leaks information about the performed computations. However, this noise can be re-randomized by re-randomizing the noise of the underlying G- and G_T-elements.

Black-box exponents. Both of our constructions use exponents only in a black-box way. Specifically, exponents are only uniformly chosen, added, and multiplied both during setup and operation of the scheme. (One subtlety here is the computation of the "reduced polynomials" $[h_i] = [X^i \bmod h]$ in the projecting and canceling construction from Section 5.4.3. Note that the coefficients of these h_i can be computed from the coefficients of h through linear operations alone. Hence, the involved exponents do not have to be explicitly divided.)

5.10.2 Using the Approximate Multilinear Maps from [3] or the GES from Chapter 3

The approximate multilinear maps from [3] and the GES from Chapter 3 do not use growing noise in encodings. Still, encodings in these constructions are randomized and efficient algorithms for, e.g., testing equality of encodings, are given in Chapter 4 of [3] and Section 3.4, respectively. With these algorithms, the constructions essentially realize the "dream version" of a multilinear map or GES, respectively, with only one difference: there is no efficient algorithm for deciding whether an encoding is valid or not. Thus, group membership is not easy to decide. As mentioned above, this can be limiting for applications but has no implications on our results.

Chapter 6

Concluding Remarks

Multilinear maps, graded encoding schemes (GES) and indistinguishability obfuscation ($i\mathcal{O}$) are cryptographic primitives that have caused excitement among the cryptographic community in the past few years. There seems to be a close relation between all of them. This thesis, for example, shows that a GES can be constructed from $i\mathcal{O}$.

The cryptographic community agrees upon multilinear maps, GESs and $i\mathcal{O}$ being powerful tools. They are eligible to answer long standing open questions of existence of cryptographic primitives in the affirmative, as also demonstrated in this thesis.

However, all existing constructions of these primitives either rely on one another or are in an uncertain state due to attacks. The bare existence of multilinear maps, GESs and $i\mathcal{O}$ is thus still being questioned. And even if this question is answered in the affirmative, the gap between theoretically efficient constructions and practically efficient ones seems to be quite big and much work has to be done to narrow it. One step towards this was made in this thesis.

Hopefully, the next years will reveal whether this exciting area of cryptography has its justification.

Bibliography

[1] Michel Abdalla, Mariana Raykova, and Hoeteck Wee. Multi-input inner-product functional encryption from pairings. Cryptology ePrint Archive, Report 2016/425, 2016. http://eprint.iacr.org/2016/425.

[2] Martin R. Albrecht, Catalin Cocis, Fabien Laguillaumie, and Adeline Langlois. Implementing candidate graded encoding schemes from ideal lattices. In Tetsu Iwata and Jung Hee Cheon, editors, *ASIACRYPT 2015, Part II*, volume 9453 of *LNCS*, pages 752–775. Springer, Heidelberg, November / December 2015.

[3] Martin R. Albrecht, Pooya Farshim, Dennis Hofheinz, Enrique Larraia, and Kenneth G. Paterson. Multilinear maps from obfuscation. In Kushilevitz and Malkin [81], pages 446–473.

[4] Diego F. Aranha, Laura Fuentes-Castañeda, Edward Knapp, Alfred Menezes, and Francisco Rodríguez-Henríquez. Implementing pairings at the 192-bit security level. In Michel Abdalla and Tanja Lange, editors, *Pairing-Based Cryptography - Pairing 2012 - 5th International Conference, Cologne, Germany, May 16-18, 2012, Revised Selected Papers*, volume 7708 of *Lecture Notes in Computer Science*, pages 177–195. Springer, 2012.

[5] Christophe Arene, Tanja Lange, Michael Naehrig, and Christophe Ritzenthaler. Faster computation of the Tate pairing. Cryptology ePrint Archive, Report 2009/155, 2009. http://eprint.iacr.org/2009/155.

[6] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In Kilian [80], pages 1–18.

[7] Mihir Bellare, Dennis Hofheinz, and Scott Yilek. Possibility and impossibility results for encryption and commitment secure under selective opening. In Antoine Joux, editor, *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 1–35. Springer, Heidelberg, April 2009.

[8] Alexandra Boldyreva, Serge Fehr, and Adam O'Neill. On notions of security for deterministic encryption, and efficient constructions without random oracles. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 335–359. Springer, Heidelberg, August 2008.

[9] Dan Boneh and Xavier Boyen. Efficient selective-ID secure identity based encryption without random oracles. In Cachin and Camenisch [25], pages 223–238.

[10] Dan Boneh and Xavier Boyen. Short signatures without random oracles. In Cachin and Camenisch [25], pages 56–73.

[11] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In Cramer [38], pages 440–456.

[12] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In Kilian [80], pages 213–229.

[13] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-DNF formulas on ciphertexts. In Joe Kilian, editor, *TCC 2005*, volume 3378 of *LNCS*, pages 325–341. Springer, Heidelberg, February 2005.

[14] Dan Boneh, Kevin Lewi, Mariana Raykova, Amit Sahai, Mark Zhandry, and Joe Zimmerman. Semantically secure order-revealing encryption: Multi-input functional encryption without obfuscation. In Oswald and Fischlin [98], pages 563–594.

[15] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. In Colin Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 514–532. Springer, Heidelberg, December 2001.

[16] Dan Boneh and Alice Silverberg. Applications of multilinear forms to cryptography. Cryptology ePrint Archive, Report 2002/080, 2002. `http://eprint.iacr.org/2002/080`.

[17] Dan Boneh and Alice Silverberg. Applications of multilinear forms to cryptography. *Contemporary Mathematics*, 324:71–90, 2003.

[18] Dan Boneh and Brent Waters. A fully collusion resistant broadcast, trace, and revoke system. In Juels et al. [77], pages 211–220.

[19] Dan Boneh and Brent Waters. Constrained pseudorandom functions and their applications. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part II*, volume 8270 of *LNCS*, pages 280–300. Springer, Heidelberg, December 2013.

[20] Dan Boneh, Brent Waters, and Mark Zhandry. Low overhead broadcast encryption from multilinear maps. In Garay and Gennaro [50], pages 206–223.

[21] Dan Boneh, David J. Wu, and Joe Zimmerman. Immunizing multilinear maps against zeroizing attacks. Cryptology ePrint Archive, Report 2014/930, 2014. `http://eprint.iacr.org/2014/930`.

[22] Xavier Boyen. The uber-assumption family (invited talk). In Steven D. Galbraith and Kenneth G. Paterson, editors, *PAIRING 2008*, volume 5209 of *LNCS*, pages 39–56. Springer, Heidelberg, September 2008.

[23] Xavier Boyen and Brent Waters. Shrinking the keys of discrete-log-type lossy trapdoor functions. In Jianying Zhou and Moti Yung, editors, *ACNS 10*, volume 6123 of *LNCS*, pages 35–52. Springer, Heidelberg, June 2010.

[24] Zvika Brakerski, Craig Gentry, Shai Halevi, Tancrède Lepoint, Amit Sahai, and Mehdi Tibouchi. Cryptanalysis of the quadratic zero-testing of ggh. Cryptology ePrint Archive, Report 2015/845, 2015. `http://eprint.iacr.org/2015/845`.

[25] Christian Cachin and Jan Camenisch, editors. *EUROCRYPT 2004*, volume 3027 of *LNCS*. Springer, Heidelberg, May 2004.

[26] Ran Canetti and Juan A. Garay, editors. *CRYPTO 2013, Part I*, volume 8042 of *LNCS*. Springer, Heidelberg, August 2013.

[27] Ran Canetti and Juan A. Garay, editors. *CRYPTO 2013, Part II*, volume 8043 of *LNCS*. Springer, Heidelberg, August 2013.

[28] Ran Canetti, Huijia Lin, Stefano Tessaro, and Vinod Vaikuntanathan. Obfuscation of probabilistic circuits and applications. In Dodis and Nielsen [40], pages 468–497.

[29] Jung Hee Cheon, Pierre-Alain Fouque, Changmin Lee, Brice Minaud, and Hansol Ryu. Cryptanalysis of the new clt multilinear map over the integers. Cryptology ePrint Archive, Report 2016/135, 2016. `http://eprint.iacr.org/2016/135`.

[30] Jung Hee Cheon, Kyoohyung Han, Changmin Lee, Hansol Ryu, and Damien Stehlé. Cryptanalysis of the multilinear map over the integers. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 3–12. Springer, Heidelberg, April 2015.

[31] Jung Hee Cheon, Jinhyuck Jeong, and Changmin Lee. An algorithm for ntru problems and cryptanalysis of the ggh multilinear map without a low level encoding of zero. Cryptology ePrint Archive, Report 2016/139, 2016. `http://eprint.iacr.org/2016/139`.

[32] Jean-Sebastien Coron, Craig Gentry, Shai Halevi, Tancrede Lepoint, Hemanta K. Maji, Eric Miles, Mariana Raykova, Amit Sahai, and Mehdi Tibouchi. Zeroizing without low-level zeroes: New mmap attacks and their limitations. Cryptology ePrint Archive, Report 2015/596, 2015. `http://eprint.iacr.org/2015/596`.

[33] Jean-Sébastien Coron, Craig Gentry, Shai Halevi, Tancrède Lepoint, Hemanta K. Maji, Eric Miles, Mariana Raykova, Amit Sahai, and Mehdi Tibouchi. Zeroizing without low-level zeroes: New MMAP attacks and their limitations. In Gennaro and Robshaw [56], pages 247–266.

[34] Jean-Sebastien Coron, Moon Sung Lee, Tancrede Lepoint, and Mehdi Tibouchi. Cryptanalysis of ggh15 multilinear maps. Cryptology ePrint Archive, Report 2015/1037, 2015. `http://eprint.iacr.org/2015/1037`.

[35] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. Practical multilinear maps over the integers. In Canetti and Garay [26], pages 476–493.

[36] Jean-Sebastien Coron, Tancrede Lepoint, and Mehdi Tibouchi. Cryptanalysis of two candidate fixes of multilinear maps over the integers. Cryptology ePrint Archive, Report 2014/975, 2014. `http://eprint.iacr.org/2014/975`.

[37] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. New multilinear maps over the integers. In Gennaro and Robshaw [56], pages 267–286.

[38] Ronald Cramer, editor. *EUROCRYPT 2005*, volume 3494 of *LNCS*. Springer, Heidelberg, May 2005.

[39] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Trans. Information Theory*, 22(6):644–654, 1976.

[40] Yevgeniy Dodis and Jesper Buus Nielsen, editors. *TCC 2015, Part II*, volume 9015 of *LNCS*. Springer, Heidelberg, March 2015.

[41] Alex Escala and Jens Groth. Fine-tuning Groth-Sahai proofs. In Hugo Krawczyk, editor, *PKC 2014*, volume 8383 of *LNCS*, pages 630–649. Springer, Heidelberg, March 2014.

[42] Alex Escala, Gottfried Herold, Eike Kiltz, Carla Ràfols, and Jorge Villar. An algebraic framework for Diffie-Hellman assumptions. In Canetti and Garay [27], pages 129–147.

[43] Pooya Farshim, Julia Hesse, Dennis Hofheinz, and Enrique Larraia. Graded encoding schemes from obfuscation. Unpublished manuscript.

[44] Marc Fischlin, Johannes Buchmann, and Mark Manulis, editors. *PKC 2012*, volume 7293 of *LNCS*. Springer, Heidelberg, May 2012.

[45] David Mandell Freeman. Converting pairing-based cryptosystems from composite-order groups to prime-order groups. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 44–61. Springer, Heidelberg, May 2010.

[46] David Mandell Freeman, Oded Goldreich, Eike Kiltz, Alon Rosen, and Gil Segev. More constructions of lossy and correlation-secure trapdoor functions. In Nguyen and Pointcheval [97], pages 279–295.

[47] Eduarda S. V. Freire, Julia Hesse, and Dennis Hofheinz. Universally composable non-interactive key exchange. In Michel Abdalla and Roberto De Prisco, editors, *SCN 14*, volume 8642 of *LNCS*, pages 1–20. Springer, Heidelberg, September 2014.

[48] Eduarda S. V. Freire, Dennis Hofheinz, Kenneth G. Paterson, and Christoph Striecks. Programmable hash functions in the multilinear setting. In Canetti and Garay [26], pages 513–530.

[49] Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Trans. Information Theory*, 31(4):469–472, 1985.

[50] Juan A. Garay and Rosario Gennaro, editors. *CRYPTO 2014, Part I*, volume 8616 of *LNCS*. Springer, Heidelberg, August 2014.

[51] Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 1–17. Springer, Heidelberg, May 2013.

[52] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th FOCS*, pages 40–49. IEEE Computer Society Press, October 2013.

[53] Sanjam Garg, Craig Gentry, Shai Halevi, Amit Sahai, and Brent Waters. Attribute-based encryption for circuits from multilinear maps. In Canetti and Garay [27], pages 479–499.

[54] Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. Witness encryption and its applications. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 467–476. ACM, 2013.

[55] Sanjam Garg, Pratyay Mukherjee, and Akshayaram Srinivasan. Obfuscation without the vulnerabilities of multilinear maps. *IACR Cryptology ePrint Archive*, 2016:390, 2016.

[56] Rosario Gennaro and Matthew J. B. Robshaw, editors. *CRYPTO 2015, Part I*, volume 9215 of *LNCS*. Springer, Heidelberg, August 2015.

[57] Craig Gentry, Sergey Gorbunov, and Shai Halevi. Graph-induced multilinear maps from lattices. In Dodis and Nielsen [40], pages 498–527.

[58] Craig Gentry, Jens Groth, Yuval Ishai, Chris Peikert, Amit Sahai, and Adam Smith. Using fully homomorphic hybrid encryption to minimize non-interative zero-knowledge proofs. *Journal of Cryptology*, pages 1–24, 2014.

[59] Craig Gentry, Shai Halevi, Hemanta K. Maji, and Amit Sahai. Zeroizing without zeroes: Cryptanalyzing multilinear maps without encodings of zero. Cryptology ePrint Archive, Report 2014/929, 2014. `http://eprint.iacr.org/2014/929`.

[60] Kristian Gjøsteen. Symmetric subgroup membership problems. In Serge Vaudenay, editor, *PKC 2005*, volume 3386 of *LNCS*, pages 104–119. Springer, Heidelberg, January 2005.

[61] Shafi Goldwasser, S. Dov Gordon, Vipul Goyal, Abhishek Jain, Jonathan Katz, Feng-Hao Liu, Amit Sahai, Elaine Shi, and Hong-Sheng Zhou. Multi-input functional encryption. In Nguyen and Oswald [96], pages 578–602.

[62] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In Juels et al. [77], pages 89–98. Available as Cryptology ePrint Archive Report 2006/309.

[63] Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In Smart [112], pages 415–432.

[64] Jens Groth and Amit Sahai. Efficient noninteractive proof systems for bilinear groups. *SIAM J. Comput.*, 41(5):1193–1232, 2012.

[65] Aurore Guillevic, François Morain, and Emmanuel Thomé. Solving discrete logarithms on a 170-bit MNT curve by pairing reduction. *CoRR*, abs/1605.07746, 2016.

[66] Shai Halevi, editor. *CRYPTO 2009*, volume 5677 of *LNCS*. Springer, Heidelberg, August 2009.

[67] Brett Hemenway, Benoît Libert, Rafail Ostrovsky, and Damien Vergnaud. Lossy encryption: Constructions from general assumptions and efficient selective opening chosen ciphertext security. In Dong Hoon Lee and Xiaoyun Wang, editors, *ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 70–88. Springer, Heidelberg, December 2011.

[68] Brett Hemenway and Rafail Ostrovsky. Extended-DDH and lossy trapdoor functions. In Fischlin et al. [44], pages 627–643.

[69] Brett Hemenway and Rafail Ostrovsky. On homomorphic encryption and chosen-ciphertext security. In Fischlin et al. [44], pages 52–65.

[70] G. Herold, J. Hesse, D. Hofheinz, C. Ràfols, and A. Rupp. Polynomial spaces: A new framework for composite-to-prime-order transformations. Cryptology ePrint Archive, 2014. `http://eprint.iacr.org/`.

[71] Gottfried Herold, Julia Hesse, Dennis Hofheinz, Carla Ràfols, and Andy Rupp. Polynomial spaces: A new framework for composite-to-prime-order transformations. In Garay and Gennaro [50], pages 261–279.

[72] Florian Hess, Nigel P. Smart, and Frederik Vercauteren. The eta pairing revisited. *IEEE Trans. Information Theory*, 52(10):4595–4602, 2006.

[73] Julia Hesse, Dennis Hofheinz, and Daniel Kraschewski. Lossy trapdoor functions with compact keys from multilinear maps. Unpublished manuscript.

[74] Julia Hesse, Dennis Hofheinz, and Andy Rupp. Reconfigurable cryptography: A flexible approach to long-term security. In Kushilevitz and Malkin [81], pages 416–445.

[75] Susan Hohenberger, Amit Sahai, and Brent Waters. Full domain hash from (leveled) multilinear maps and identity-based aggregate signatures. In Canetti and Garay [26], pages 494–512.

[76] Antoine Joux. A one round protocol for tripartite Diffie-Hellman. *Journal of Cryptology*, 17(4):263–276, September 2004.

[77] Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors. *ACM CCS 06*. ACM Press, October / November 2006.

[78] A. Karatsuba and Yu. Ofman. Multiplication of many-digital numbers by automatic computers. In *USSR Academy of Sciences*, volume 145, pages 293–294, 1962.

[79] Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In Smart [112], pages 146–162.

[80] Joe Kilian, editor. *CRYPTO 2001*, volume 2139 of *LNCS*. Springer, Heidelberg, August 2001.

[81] Eyal Kushilevitz and Tal Malkin, editors. *TCC 2016-A, Part I*, volume 9562 of *LNCS*. Springer, Heidelberg, January 2016.

[82] Adeline Langlois, Damien Stehlé, and Ron Steinfeld. GGHLite: More efficient multilinear maps from ideal lattices. In Nguyen and Oswald [96], pages 239–256.

[83] Hyang-Sook Lee. A self-pairing map and its applications to cryptography. *Applied Mathematics and Computation*, 151(3):671–678, 2004.

[84] Kevin Lewi, Alex J. Malozemoff, Daniel Apon, Brent Carmer, Adam Foltzer, Daniel Wagner, David W. Archer, Dan Boneh, Jonathan Katz, and Mariana Raykova. 5gen: A framework for prototyping applications using multilinear maps and matrix branching programs. Cryptology ePrint Archive, Report 2016/619, 2016. `http://eprint.iacr.org/2016/619`.

[85] Allison B. Lewko. Tools for simulating features of composite order bilinear groups in the prime order setting. In Pointcheval and Johansson [100], pages 318–335.

[86] Allison B. Lewko and Sarah Meiklejohn. A profitable sub-prime loan: Obtaining the advantages of composite-order in prime-order bilinear groups. *IACR Cryptology ePrint Archive*, 2013:300, 2013.

[87] Allison B. Lewko and Sarah Meiklejohn. A profitable sub-prime loan: Obtaining the advantages of composite order in prime-order bilinear groups. In Jonathan Katz, editor, *PKC 2015*, volume 9020 of *LNCS*, pages 377–398. Springer, Heidelberg, March / April 2015.

[88] R. J. McEliece. A Public-Key Cryptosystem Based On Algebraic Coding Theory. *Deep Space Network Progress Report*, 44:114–116, January 1978.

[89] Sarah Meiklejohn, Hovav Shacham, and David Mandell Freeman. Limitations on transformations from composite-order to prime-order groups: The case of round-optimal blind signatures. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 519–538. Springer, Heidelberg, December 2010.

[90] Alfred Menezes, Scott A. Vanstone, and Tatsuaki Okamoto. Reducing elliptic curve logarithms to logarithms in a finite field. In Cris Koutsougeras and Jeffrey Scott Vitter, editors, *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing, May 5-8, 1991, New Orleans, Louisiana, USA*, pages 80–89. ACM, 1991.

[91] R. Merkle and M. Hellman. Hiding information and signatures in trapdoor knapsacks. *IEEE Trans. Inf. Theor.*, 24(5):525–530, September 2006.

[92] Ralph C. Merkle. Secure communications over insecure channels. *Commun. ACM*, 21(4):294–299, 1978.

[93] Eric Miles, Amit Sahai, and Mark Zhandry. Annihilation attacks for multilinear maps: Cryptanalysis of indistinguishability obfuscation over GGH13. *IACR Cryptology ePrint Archive*, 2016:147, 2016.

[94] Petros Mol and Scott Yilek. Chosen-ciphertext security from slightly lossy trapdoor functions. In Nguyen and Pointcheval [97], pages 296–311.

[95] Moni Naor and Gil Segev. Public-key cryptosystems resilient to key leakage. In Halevi [66], pages 18–35.

[96] Phong Q. Nguyen and Elisabeth Oswald, editors. *EUROCRYPT 2014*, volume 8441 of *LNCS*. Springer, Heidelberg, May 2014.

[97] Phong Q. Nguyen and David Pointcheval, editors. *PKC 2010*, volume 6056 of *LNCS*. Springer, Heidelberg, May 2010.

[98] Elisabeth Oswald and Marc Fischlin, editors. *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*. Springer, Heidelberg, April 2015.

[99] Chris Peikert and Brent Waters. Lossy trapdoor functions and their applications. In Richard E. Ladner and Cynthia Dwork, editors, *40th ACM STOC*, pages 187–196. ACM Press, May 2008.

[100] David Pointcheval and Thomas Johansson, editors. *EUROCRYPT 2012*, volume 7237 of *LNCS*. Springer, Heidelberg, April 2012.

[101] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, 1978.

[102] Julia Rohlfing. Paarungen auf elliptischen kurven und ihre anwendung in der kryptografie. Diplomarbeit, Universität Karlsruhe (TH), Germany, April 2009.

[103] Alon Rosen and Gil Segev. Chosen-ciphertext security via correlated products. In Omer Reingold, editor, *TCC 2009*, volume 5444 of *LNCS*, pages 419–436. Springer, Heidelberg, March 2009.

[104] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In David B. Shmoys, editor, *46th ACM STOC*, pages 475–484. ACM Press, May / June 2014.

[105] Amit Sahai and Brent R. Waters. Fuzzy identity-based encryption. In Cramer [38], pages 457–473.

[106] Ryuichi Sakai and Masao Kasahara. ID based cryptosystems with pairing on elliptic curve. *IACR Cryptology ePrint Archive*, 2003:54, 2003.

[107] Jae Hong Seo. On the (im)possibility of projecting property in prime-order setting. In Xiaoyun Wang and Kazue Sako, editors, *ASIACRYPT 2012*, volume 7658 of *LNCS*, pages 61–79. Springer, Heidelberg, December 2012.

[108] Jae Hong Seo and Jung Hee Cheon. Beyond the limitation of prime-order bilinear groups, and round optimal blind signatures. In Ronald Cramer, editor, *TCC 2012*, volume 7194 of *LNCS*, pages 133–150. Springer, Heidelberg, March 2012.

[109] Peter W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *35th Annual Symposium on Foundations of Computer Science, Santa Fe, New Mexico, USA, 20-22 November 1994*, pages 124–134. IEEE Computer Society, 1994.

[110] Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 256–266. Springer, Heidelberg, May 1997.

[111] Joseph H. Silverman. *The arithmetic of elliptic curves*. Graduate texts in mathematics. Springer, New York, Berlin, 1986.

[112] Nigel P. Smart, editor. *EUROCRYPT 2008*, volume 4965 of *LNCS*. Springer, Heidelberg, April 2008.

[113] Frederik Vercauteren. Optimal pairings. *IEEE Trans. Information Theory*, 56(1):455–461, 2010.

[114] Brent Waters. Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions. In Halevi [66], pages 619–636.

[115] Hoeteck Wee. Dual projective hashing and its applications - lossy trapdoor functions and more. In Pointcheval and Johansson [100], pages 246–262.

[116] André Weimerskirch and Christof Paar. Generalizations of the Karatsuba algorithm for efficient implementations. Cryptology ePrint Archive, Report 2006/224, 2006. `http://eprint.iacr.org/`.

[117] Alberto Zanoni. Iterative Karatsuba method for multivariate polynomial multiplication. In *Proceedings of the International Conference on Theory and Applications of Mathematics and Informatics, ICTAMI 2009*, pages 829–843, September 2009.

[118] Fangguo Zhang, Reihaneh Safavi-Naini, and Willy Susilo. An efficient signature scheme from bilinear pairings and its applications. In Feng Bao, Robert Deng, and Jianying Zhou, editors, *PKC 2004*, volume 2947 of *LNCS*, pages 277–290. Springer, Heidelberg, March 2004.

[119] Joe Zimmerman. How to obfuscate programs directly. In Oswald and Fischlin [98], pages 439–467.

List of Figures

List of Tables