Graded Encoding Schemes from Obfuscation[*]

Pooya Farshim[†,1,2], Julia Hesse[1,2,3], Dennis Hofheinz[‡,3], and Enrique Larraia[§]

[1]DIENS, École normale supérieure, CNRS, PSL Research University, Paris, France
[2]INRIA
[3]Karlsruhe Institute of Technology, Germany

{pooya.farshim, julia.hesse}@ens.fr,
dennis.hofheinz@kit.edu, elarraia@gmail.com

April 12, 2018

Abstract

We construct a graded encoding scheme (GES), an approximate form of graded multilinear maps. Our construction relies on indistinguishability obfuscation, and a pairing-friendly group in which (a suitable variant of) the strong Diffie–Hellman assumption holds. As a result of this abstract approach, our GES has a number of advantages over previous constructions. Most importantly:

- We can *prove* that the multilinear decisional Diffie–Hellman (MDDH) assumption holds in our setting, assuming the used ingredients are secure (in a well-defined and standard sense). Hence, our GES does not succumb to so-called "zeroizing" attacks if the underlying ingredients are secure.
- Encodings in our GES do not carry any noise. Thus, unlike previous GES constructions, there is no upper bound on the number of operations one can perform with our encodings. Hence, our GES essentially realizes what Garg et al. (EUROCRYPT 2013) call the "dream version" of a GES.

Technically, our scheme extends a previous, non-graded approximate multilinear map scheme due to Albrecht et al. (TCC 2016-A). To introduce a graded structure, we develop a new view of encodings at different levels as polynomials of different degrees.

Keywords. Multilinear maps, graded encoding schemes, indistinguishability obfuscation.

[*]An extended abstract of this work appears in the proceedings of PKC 2018. This is the full version.
[†]Supported in part by grant ANR-14-CE28-0003 (Project EnBid)
[‡]Supported by ERC grant 724307, and by DFG grants HO 4534/2-2 and HO 4534/4-1.
[§]Work done white at Royal Holloway, University of London, United Kingdom, supported by EPSRC grant EP/L018543/1.

Contents

1 Introduction

The GGH candidate multilinear map. In 2013, Garg, Gentry, and Halevi (GGH) [GGH13a] proposed the first plausible construction of an (approximate) multilinear map (MLM). In a nutshell, an MLM is a map $e : \mathbb{G}^\kappa \longrightarrow \mathbb{G}_T$ (for groups $\mathbb{G}$ and $\mathbb{G}_T$) that is linear in each input. Of course, we are most interested in the case of "cryptographically interesting" groups $\mathbb{G}$ (in which, e.g., computing discrete logarithms is infeasible), non-trivial maps e (with non-trivial kernel), and preferably large values of κ. The surprising cryptographic consequences of such "cryptographically interesting" MLMs were already investigated in 2003 by Boneh and Silverberg [BS03], but an actual construction of an MLM remained elusive until the candidate construction of GGH.

Unfortunately, GGH only presented an "approximate" MLM in the following sense:

- Instead of group elements, their e inputs (and outputs) are *encodings*. An encoding is a non-unique representation of a group element, and there is no guarantee about which particular encoding the group operation (or e) outputs. However, every encoding allows to derive a "canonical form" that uniquely determines the encoded group element. (This canonical form allows no further operations, though.)
- Each encoding carries a "noise level" that increases with each operation. If the noise level grows beyond a certain threshold, no further operations are possible.

However, the GGH MLM also has an important *graded* property that allows to evaluate e partially, in a sense we will detail later. In particular this graded structure has made the GGH MLM tremendously useful: notable applications of *graded* MLMs include indistinguishability obfuscation [GGH+13b], witness encryption [GGSW13], attribute-based encryption for general circuits [GGH+13c], and constrained pseudorandom functions for general circuits [BW13]. Furthermore, graded MLMs enable a very powerful class of programmable hash functions [HK08], which in turn allows to implement random oracles in certain "algebraic" applications [HSW13, FHPS13].

After GGH's MLM construction, several other (graded and approximate) MLM constructions have been proposed [CLT13, LSS14, GGH15, CLT15]. However, *all* of these constructions (including the original GGH scheme) succumb to cryptanalytic attacks [CHL+15, CGH+15, CLLT16, MSZ16]. In particular, currently there is no obvious way to instantiate schemes relying on multilinear maps, e.g., the schemes from [GGSW13, GGH+13c, BW13, HSW13, FHPS13].[1]

Graded MLMs. There is one (approximate) MLM construction of Albrecht, Farshim, Hofheinz, Larraia, and Paterson (AFHLP) [AFH+16] that does not fall victim to any of the mentioned cryptanalytic attacks on MLMs. However, this construction does not offer a *graded* MLM, and thus cannot be used to bootstrap, e.g., witness encryption. Graded MLMs are algebraic tools that can enable other algebraic tools such as multilinear Groth-Sahai proofs, or multilinear programmable hash functions. It is thus still an interesting open problem whether graded MLMs exist, and whether the results of [GGH+13b] can be augmented to even show equivalence to indistinguishability obfuscation.

Our contribution. In this work, we construct graded, approximate MLMs that do not succumb to any of the known attacks. Technically, we extend the non-graded MLM construction from AFHLP [AFH+16] to a graded MLM. We prove that the multilinear decisional Diffie–Hellman (MDDH) assumption [GGH13a] holds relative to our MLM, provided that the used ingredients are secure.

Interestingly, our MLM has two technical features that previous graded approximate MLMs do not have:

1. Our encodings do not carry any noise (although they are not unique). In particular, there is no limit on the number of operations that one can perform with our encodings.
2. The canonical forms derived from encodings allow further group operations (but no further pairings).

Our new MLM (when implemented with the indistinguishability obfuscator from [GGH+13b, GMS16]) currently forms the only plausible graded MLM, and thus the only plausible way to implement a number of MLM-based constructions [GGSW13, GGH+13c, BW13, HSW13, FHPS13].

[1]We note, however, that the cryptographic *tasks* that the constructions from [GGSW13, BW13] aim to achieve can be directly achieved with indistinguishability obfuscation [GGH+13b, SW14, AFP16].

Furthermore, our construction is generic and modular. In particular, we reduce the quest to develop a secure (graded) MLM 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 MLM candidates (which were based on complex algebraic or combinatorial assumptions), this is not an "understood feature" at all for MLMs.

In fact, taken together with recent constructions of indistinguishability obfuscation (iO) from multilinear maps (e.g., [GGH+13b, Lin16, AS17, LT17]), our result shows a (somewhat loose) equivalence of indistinguishability obfuscation (iO) and (graded and approximate) MLMs, in the presence of a pairing-friendly group. This equivalence is loose in the following sense. First, the assumptions on both ends of the equivalence do not match: some of these works (e.g., [GGH+13b]) construct iO from MLMs which support very strong computational assumptions (much stronger than MDDH) or require asymmetric multilinear maps. On the other hand, we use iO to construct *symmetric* MLMs 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, or to adapt our construction to asymmetric pairings, and we leave open to tighten this equivalence. Second, going through our equivalence suffers subexponential security loss. Namely, we require *probabilistic* indistinguishability obfuscation, which can be constructed from iO [CLTV15], but currently only through a sub-exponential reduction.

However, we note that such an equivalence would not be highly surprising given recent results on constructing iO from MLMs [Lin16, AS17]. These works only require "one-shot" (but asymmetric) MLMs, and not even *graded* encodings as we construct them.

Related Work. Our work is closely related to [AFH+16], since the non-graded MLM there serves as a starting point for our graded MLM. We will summarize their construction in Section 4 and give an informal overview below.

Recently, Paneth and Sahai [PS15] have shown a near-equivalence of a suitable abstraction of MLMs with iO. Their result requires no computational assumptions at all, but also does not consider MLMs in our sense. In particular, they construct an abstraction of a MLM that only admits restricted access to encodings similar to the one in [GGH+13b]. Beyond the group operation and the multilinear map, efficient procedures for, e.g., uniform sampling, comparison or rerandomization of encodings, are not part of this abstraction. Conversely, our notion of a MLM, like the ones from [AFH+16, GGH13a], contains descriptions of efficient procedures for these tasks.

It would be interesting to see how the restricted MLMs of [PS15] can be used to instantiate the constructions from [FHPS13, HSW13, BWZ14, BLR+15] directly, i.e., without making the detour via iO. However, since iO alone is not even known to imply one-way functions (see [GR07] for a discussion), this will require additional assumptions.

Pass et al. [PST14] give a security definition of graded MLMs that requires that whenever encodings are generically equivalent (that is, cannot be distinguished with generic operations alone), they should be computationally indistinguishable as encodings. They show that this MLMs which satisfy this strong assumption imply indistinguishability obfuscation. It is not clear, however, how to construct such strongly secure MLMs (without resorting to idealized models such as the generic group model).

1.1 The (non-graded) approximate multilinear map of AFHLP

Encodings. Since our own construction is an extension of the (non-graded) approximate MLM of [AFH+16], we first recall their work. Simplifying slightly, AFHLP encode a group element g^z (from a cyclic group $\mathbb{G}$ of order p) as

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

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 [AFH+16] 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

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

where "$*$" denotes the convolution product of vectors.[2] The values g^{ω^i} (for $i \leq \kappa$) are hardwired into **Mult**, so **Mult** can compute $g^{\prod z_i}$ through (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.)

1.2 Our new graded encoding scheme

Before proceeding any further, we briefly recall the notions of a graded multilinear map and a graded encoding scheme.

Graded maps. In a *graded* multilinear map setting, we have groups $\mathbb{G}_1, \ldots, \mathbb{G}_\kappa$, and (efficiently computable) bilinear maps $e_{i,j} : \mathbb{G}_i \times \mathbb{G}_j \longrightarrow \mathbb{G}_{i+j}$ for $i + j \leq \kappa$. Hence, the $e_{i,j}$ also allow the evaluation of a multilinear map $e : \mathbb{G}_1^\kappa \longrightarrow \mathbb{G}_\kappa$ iteratively, e.g., through

$$e(g_1, \ldots, g_\kappa) \ := \ e_{1,\kappa-1}(g_1, e_{1,\kappa-2}(g_2, \cdots, e_{1,1}(g_{\kappa-1}, g_\kappa) \cdots)) \ .$$

However, the $e_{i,j}$ also allow "partial" evaluation of e, which is the key to entirely new applications such as those in [GGH+13b, GGSW13, GGH+13c, BW13].

Unfortunately, we do not currently know how to implement such a "clean" graded multilinear map. Instead, all known graded MLM constructions work on encodings (i.e., non-unique representations of group elements). Such a construction is usually called a graded encoding scheme (GES). Following the GES notation, we will henceforth also call an encoding of a $\mathbb{G}_\ell$-element a *level-ℓ encoding*.

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

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

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

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

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 $\mathbf{Mult}_{i,j}$. Circuit $\mathbf{Mult}_{i,j}$ 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 $\mathbf{Mult}_{i,j}$ is a level-$(i+j)$ encoding $h = (g^z, c, \pi, i+j)$, computed as follows:[3]

- 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

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

  (Since the u_ℓ 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 MLM. Namely, observe that any $\mathbf{Mult}_{i,j}$ can only multiply encodings if $i + j \leq \kappa$. To compute the first component g^z of any "higher-level" encoding, knowledge of g^{ω^ℓ} for $\ell > i + j$ seems to be required. 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.)

[3] Since $\mathbf{Mult}_{i,j}$ 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 (see [GGH13a, Appendix A] for a definition).

To this end, we will use *double encryptions* c (instead of the single encryption $c = \mathbf{Enc}(P, pk)$ 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 $\mathbf{Mult}_{i,j}$ output *encodings* (and not just group elements as in the multilinear map of AFHLP). Hence, our $\mathbf{Mult}_{i,j}$ circuits also need to construct consistency proofs π, which requires additional secrets (as witnesses) in the description of $\mathbf{Mult}_{i,j}$ 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 and tighter proofs.

Fortunately, the indistinguishability obfuscator from [GGH$^+$13b] requires only a relatively weak MLM variant and hence is not affected by the above-mentioned cryptanalyses.[4]

Assumptions. In summary, our construction uses a cyclic group in which the SDDH assumption holds, a probabilistic indistinguishability obfuscation scheme [CLTV15], a perfectly correct fully homomorphic encryption (FHE), a dual-mode non-interactive zero-knowledge proof systems, and a language with hard membership. All of these assumptions are implied by pairing-friendly SDDH groups (equipped with an asymmetric pairing) and sub-exponentially secure indistinguishability obfuscation (see [GS12]). We stress that plausible candidates for both ingredients exist (e.g., by combining [GGH13a] and [GGH$^+$13b] to an indistinguishability obfuscator candidate).

Road map. We first recall some preliminaries in Section 2, the GES definition in Section 3, and the AFHLP construction in Section 4. Then, we present our GES construction in Section 5, and establish our central technical tool (the "switching theorem") in Section 6. We prove the hardness of the MDDH in Section 7. In the appendices we provide the full proofs.

2 Preliminaries

Notation. We denote the security parameter by $\lambda \in \mathbb{N}$ and assume that it is implicitly given to all algorithms in the unary representation 1^λ.

By an algorithm we mean a stateless Turing machine. Algorithms are randomized unless stated otherwise, and PPT as usual stands for "probabilistic polynomial-time." In this paper, by a PPT algorithm we mean an algorithm that runs in polynomial time in the security parameter (rather than the total length of its inputs).

Given a randomized algorithm $\mathcal{A}$ we denote the action of running $\mathcal{A}$ on input(s) $(1^\lambda, x_1, \ldots)$ with uniform random coins r and assigning the output(s) to $(y_1, \ldots)$ by $(y_1, \ldots) \leftarrow_{\$} \mathcal{A}(1^\lambda, x_1, \ldots; r)$.

For a finite set X, we denote its cardinality by $|X|$ and the action of sampling a uniformly random element x from X by $x \leftarrow_{\$} X$.

Similarly, for a finite set X, the action of sampling a uniformly random element x is denoted by $x \leftarrow_{\$} X$.

We write $[k] := \{1, \ldots, k\}$. Vectors are written in boldface $\mathbf{x}$, and slightly abusing notation, running algorithms on vectors of elements indicates component-wise operation.

Throughout the paper $\perp$ denotes a special error symbol, and $\mathrm{poly}(\cdot)$ stands for a fixed (but unspecified) polynomial.

A real-valued function $\mathrm{negl}(\lambda)$ is negligible if $\mathrm{negl}(\lambda) \in \mathcal{O}(\lambda^{-\omega(1)})$. We denote the set of all negligible functions by Negl. We use bracket notation for elements in $\mathbb{G}$, i.e., writing $[z]$ and $[z']$ for two elements g^z

[4] A recent attack on MLMs (see [MSZ16]) tackles even the weak MLM security requirements the indistinguishability obfuscator from [GGH$^+$13b] has. However, the construction of [GGH$^+$13b] (resp., its MLM building block) can be suitably enhanced to thwart this attack [GMS16].

and $g^{z'}$ in $\mathbb{G}$ and $[z]+[z']$ for their product $g^z g^{z'}$.

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 poly(λ)-sized deterministic circuits (for a fixed polynomial poly(λ)).

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 $\mathrm{C}(x;r)$.

2.1 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 $\Pi := (\mathbf{Gen}, \mathbf{Enc}, \mathbf{Dec}, \mathbf{Eval})$ such that $(\mathbf{Gen}, \mathbf{Enc}, \mathbf{Dec})$ 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. For the scheme $\Pi := (\mathbf{Gen}, \mathbf{Enc}, \mathbf{Dec})$, we require *perfect* correctness as a PKE scheme; that is, for any $\lambda \in \mathbb{N}$, any $m \in \{0,1\}^\lambda$, any $(sk, pk) \leftarrow\!\!\$\ \mathbf{Gen}(1^\lambda)$, and any $c \leftarrow\!\!\$\ \mathbf{Enc}(m, pk)$ we have that $\mathbf{Dec}(c, sk) = m$. 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 $(sk, pk) \leftarrow\!\!\$\ \mathbf{Gen}(1^\lambda)$ and any $c_i \leftarrow\!\!\$\ \mathbf{Enc}(m_i, pk)$ we have that $\mathbf{Eval}(pk, \mathrm{C}, c_1, \ldots, c_n)$ is in the range of $\mathbf{Enc}(\mathrm{C}(m_1, \ldots, m_n), pk)$.

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.

A perfectly correct FHE scheme can be constructed from probabilistic indistinguishability obfuscation (and a re-randomizable public-key encryption scheme such as ElGamal), see [CLTV15]. (We note that the FHE scheme from [CLTV15] only enjoys perfect correctness when the obfuscator and encryption scheme are also perfectly correct.)

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 1 (left). Adversary $\mathcal{A}$ is legitimate if it outputs two messages of equal lengths.

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

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

```
$\underline{\text{IND}^{\mathcal{A}}_{\mathbf{Obf}}(\lambda)}$:
  $(\mathrm{C}_0, \mathrm{C}_1, st) \leftarrow\$ \mathcal{A}_1(1^\lambda)$
  $b \leftarrow\$ \{0,1\}$
  $\overline{\mathrm{C}} \leftarrow\$ \mathbf{Obf}(1^\lambda, \mathrm{C}_b)$
  $b' \leftarrow\$ \mathcal{A}_2(\overline{\mathrm{C}}, st)$
  Return $(b = b')$
```

```
$\underline{\text{Sel-IND}^{\mathcal{D}}_{\mathcal{A}}(\lambda)}$:
  $(x, z) \leftarrow\$ \mathcal{D}_1(1^\lambda)$
  $(\mathrm{C}_0, \mathrm{C}_1, st) \leftarrow\$ \mathcal{A}_1(1^\lambda)$
  $b \leftarrow\$ \{0,1\}$; $r \leftarrow\$ \{0,1\}^{\mathrm{rl}(\lambda)}$
  $y \leftarrow \mathrm{C}_b(x; r)$
  $b' \leftarrow\$ \mathcal{D}_2(y, \mathrm{C}_0, \mathrm{C}_1, st, z)$
  Return $(b = b')$
```

Figure 1: **Left**: IND-CPA security of a (homomorphic) PKE scheme. **Middle**: 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 $\mathcal{A} := (\mathcal{A}_1, \mathcal{A}_2)$.

equivalent; that is, that for all $\lambda \in \mathbb{N}$, all $\mathrm{C} \in \mathcal{C}_\lambda$, all $\overline{\mathrm{C}} \leftarrow\$ \mathbf{Obf}(1^\lambda, \mathrm{C})$, 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 [CLTV15] 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 $\overline{\mathrm{C}} \leftarrow\$ \mathbf{Obf}(1^\lambda, \mathrm{C})$ 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 [CLTV15] 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 $\mathcal{A} := (\mathcal{A}_1, \mathcal{A}_2)$

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

where game IND is shown in Figure 1 (middle). Depending on the adopted notion of legitimacy, different security notions for the obfuscator emerge; we consider the following one.

X-IND samplers [CLTV15]. 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 hold: (1) For all (even unbounded) $\mathcal{D}$ the advantage function below is negligible.

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

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

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

where game $\text{Sel-IND}^{\mathcal{D}}_{\mathcal{A}}(\lambda)$ is shown in Figure 1 (right).This game is named "static-input-IND" in [CLTV15]. 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* [CLTV15] 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.

2.3 Dual-mode NIZK proof systems

In our constructions we will be relying 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 [GS08], and using a generic NP reduction to the satisfiability of quadratic equations we can obtain a suitable proof system for any NP language.[5] 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) $\mathbf{BCRS}(gpk)$ on input gpk in the support of $\mathbf{G}$ outputs a (binding) CRS crs and an extraction trapdoor td_{ext}; (2) $\mathbf{HCRS}(gpk)$ on input gpk in the support of $\mathbf{G}$ outputs a (hiding) CRS crs and a simulation trapdoor td_{zk}; (3) $\mathbf{Prove}(gpk, crs, x, w)$ on input gpk a first coordinate in the support of $\mathbf{G}$, a CRS crs, an instance x, and a witness w, outputs a proof π; (4) $\mathbf{Verify}(gpk, crs, x, \pi)$ on input gpk, crs, an instance x, and a proof π, outputs 1 for accept or 0 for reject; (5) $\mathbf{WExt}(td_{ext}, x, \pi)$ on input an extraction trapdoor td_{ext}, an instance x, and a proof π, outputs a witness w; and (6) $\mathbf{Sim}(td_{zk}, x)$ 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 CRSs generated with $\mathbf{BCRS}(gpk)$ and $\mathbf{HCRS}(gpk)$ are computationally indistinguishable. Formally, we require the advantage of any PPT adversary $\mathcal{A}$ defined below to be negligible.

$$\begin{aligned}\mathbf{Adv}^{\mathrm{crs}}_{\Sigma,\mathcal{A}}(\lambda) := 2\cdot \Pr\big[&b \leftarrow_\$ \{0,1\}; gpk \leftarrow_\$ \mathbf{G}(1^\lambda); (crs_0, td_{ext}) \leftarrow_\$ \mathbf{BCRS}(gpk);\\ &(crs_1, td_{zk}) \leftarrow_\$ \mathbf{HCRS}(gpk); b' \leftarrow_\$ \mathcal{A}(gpk, crs_b) : b = b'\big] - 1\end{aligned}$$

Perfect completeness. For any $\lambda \in \mathbb{N}$, any $gpk \leftarrow_\$ \mathbf{G}(1^\lambda)$, any $(crs, td_{ext}) \leftarrow_\$ \mathbf{BCRS}(gpk)$, any (x, w) where it holds that $\mathbf{R}(gpk, x, w) = 1$, and any $\pi \leftarrow_\$ \mathbf{Prove}(gpk, crs, x, w)$, it holds that $\mathbf{Verify}(gpk, crs, x, \pi) = 1$. We require this property to also hold for any choice of hiding CRS.

Perfect soundness under $\mathbf{BCRS}$. For any $\lambda \in \mathbb{N}$, any $gpk \leftarrow_\$ \mathbf{G}(1^\lambda)$, any CRS $(crs, td_{ext}) \leftarrow_\$ \mathbf{BCRS}(gpk)$, any x where it holds that $\mathbf{R}(gpk, x, w) = 0$ for all $w \in \{0,1\}^*$, and any $\pi \in \{0,1\}^*$ we have that $\mathbf{Verify}(gpk, crs, x, \pi) = 0$.

Perfect extraction under $\mathbf{BCRS}$. For any $\lambda \in \mathbb{N}$, any $gpk \leftarrow_\$ \mathbf{G}(1^\lambda)$, any CRS $(crs, td_{ext}) \leftarrow_\$ \mathbf{BCRS}(gpk)$, any (x, π) with $\mathbf{Verify}(gpk, crs, x, \pi) = 1$, and any $w \leftarrow_\$ \mathbf{WExt}(td_{ext}, x, \pi)$ we have that $\mathbf{R}(gpk, x, w) = 1$.

Perfect Witness Indistinguishability under $\mathbf{HCRS}$. For any $\lambda \in \mathbb{N}$, any $gpk \leftarrow_\$ \mathbf{G}(1^\lambda)$, any $(crs, td_{zk}) \leftarrow_\$ \mathbf{HCRS}(gpk)$, and any (x, w_b) such that $\mathbf{R}(gpk, x, w_b) = 1$ for $b \in \{0,1\}$, the two distributions $\pi_b \leftarrow_\$ \mathbf{Prove}(gpk, crs, x, w_b)$ are identical.

Perfect Zero Knowledge under $\mathbf{HCRS}$. For any $\lambda \in \mathbb{N}$, any $gpk \leftarrow_\$ \mathbf{G}(1^\lambda)$, any $(crs, td_{zk}) \leftarrow_\$ \mathbf{HCRS}(gpk)$, and any (x, w) such that $\mathbf{R}(gpk, x, w) = 1$, the two distributions $\pi_0 \leftarrow_\$ \mathbf{Prove}(gpk, crs, x, w)$ and $\pi_1 \leftarrow_\$ \mathbf{Sim}(td_{zk}, x)$ are identical.

2.4 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 tuple of four algorithms

[5]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. (Extraction will recover g^{w_i} for all i, and thus all w_i too.)

$\Lambda := (\mathbf{Gen_L}, \mathbf{YesSam_L}, \mathbf{NoSam_L}, \mathbf{R_L})$ as follows. (1) $\mathbf{Gen_L}(1^\lambda)$ is randomized and on input the security parameter outputs a language key lk; (2) $\mathbf{YesSam_L}(lk)$ is randomized and on input the language key lk outputs a yes-instance y; (3) $\mathbf{NoSam_L}(lk)$ is randomized and on input the language key lk outputs a no-instance y; and (4) $\mathbf{R_L}(lk, y, w)$ is deterministic and on input lk, an instance y and a witness w outputs 1 for true or 0 for false.

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

Finally, the language is required to have a hard membership problem in the sense that for any PPT adversary $\mathcal{A}$

$$\mathbf{Adv}^{\mathrm{mem}}_{\Lambda,\mathcal{A}}(\lambda) := 2\cdot \Pr\big[b \leftarrow_\$ \{0,1\}; lk \leftarrow_\$ \mathbf{Gen_L}(1^\lambda); y_0 \leftarrow_\$ \mathbf{NoSam_L}(lk); \\ y_1 \leftarrow_\$ \mathbf{YesSam_L}(lk); b' \leftarrow_\$ \mathcal{A}(lk, y_b) : b = b'\big] - 1 \in \textsc{Negl} .$$

Such languages can be instantiated using the DDH problem as follows. Algorithm $\mathbf{Gen_L}(1^\lambda)$ outputs the description of a prime-order group $(\mathbb{G}, g, p, 1)$ as lk. Algorithm $\mathbf{YesSam_L}(lk)$ samples a Diffie–Hellman tuple (g^a, g^b, g^{ab}), and $\mathbf{NoSam_L}(lk)$ outputs a non-Diffie–Hellman tuple (g^a, g^b, g^c) for a random $c \neq ab \pmod p$ when $b = 0$. Relation $\mathbf{R_L}$ 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 Graded Encoding Schemes

We start by recalling (a slight variant of) the definition of graded encoding systems from Garg, Gentry and Halevi (GGH) [GGH13a].

κ-GRADED ENCODING SYSTEM. Let R be a (non-trivial) commutative ring and $S := \{S_i^{(a)} \subset \{0,1\}^* : a \in R, 0 \le i \le \kappa\}$ 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 $i + j \le \kappa$, there is a binary operation "×" 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 "×" 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 "×" 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 $S_j^{(a_1+a_2+a_3)}$.

Throughout the paper, 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.

κ-GRADED ENCODING SCHEME. Let (R, S) be a κ-graded encoding system. A *graded encoding scheme (GES)*

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

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

$\mathbf{Setup}(1^\lambda, 1^\kappa)$: 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 poly(λ) as long as κ is polynomial in λ.

$\mathbf{Eq}_i(h_1, h_2)$: For $i \in \{0, \dots, \kappa\}$ 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.

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

$\mathbf{Mult}_{i,j}(h_1, h_2)$: This deterministic algorithm performs the "$\times$" operation of (R, S). For $i, j \in \{0, \dots, \kappa\}$ 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 $S_{i+j}^{(a_1 \cdot a_2)}$.

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

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

Our definition of a GES essentially implements the "dream version" of GESs [GGH13a], 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 $\mathbf{Eq}_i$) 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.

4 Approximate Multilinear Maps

We recall the approximate multilinear maps due to AFHLP [AFH$^+$16]. The authors construct both symmetric and asymmetric multilinear maps. Their symmetric construction can be seen as a starting point for our GES.

4.1 Syntax

We start with the syntax of multilinear group (MLG) schemes [AFH$^+$16]. 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 in terms of a GES.

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.

4.2 Overview of AFHLP

In a nutshell, [AFH$^+$16] 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 $\mathbf{x} = (x_0, x_1)$ *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 $\mathbf{x}$ representing h, ciphertext c_2 is a homomorphic encryption under a second public key pk_2 of another vector $\mathbf{y}$ also representing h, and π is a NIZK proof showing consistency of the two vectors $\mathbf{x}$ and $\mathbf{y}$. Here consistency means that the plaintexts vectors $\mathbf{x}$ and $\mathbf{y}$ 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. 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 witness (sk_1, sk_2) to generate a NIZK proof showing equality of encodings. Note that the new encoding is as compact as the two input encodings.

The multilinear map on inputs $(h_i, c_{i,1}, c_{i,2}, \pi_i)$ for $1 \leq i \leq \kappa$ is computed using an obfuscation of a circuit $\mathrm{C}_{\mathrm{Map}}[sk_1, \omega]$, which has sk_1 and ω hardwired in. 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 $\mathbf{Dec}(\cdot, sk_1)$. 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. See Appendix A for technical details.

In the original paper, this construction is generalized to the asymmetric setting via representations of the form $g^{\langle \mathbf{x}, \boldsymbol{\omega} \rangle}$ with $\mathbf{x}, \boldsymbol{\omega} \in \mathbb{Z}_N^\ell$ for $\ell \in \{2, 3\}$ (where $\langle \mathbf{x}, \boldsymbol{\omega} \rangle$ denotes inner products modulo the base-group order). The special case $\boldsymbol{\omega} := (1, \omega)$ then gives an MLG scheme where MDDH is shown to be hard. We refer the reader to the original work [AFH$^+$16] for the details.

5 The GES Construction

We now present our construction of a graded encoding scheme Γ according to the syntax introduced in Section 3.

We will use the following ingredients in our construction. A similar set of building blocks were used in [AFH$^+$16].

1. A group setup algorithm $\mathbf{Setup}_{\mathbb{G}}(1^\lambda)$ 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.[6] We implicitly assume efficient algorithms for checking group membership, performing the group operation, inversion, and randomly sampling group elements. We further assume a unique binary representation for every group element and 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.

[6] 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.

5. A language family Λ with hard membership problem and unique witnesses.

Given the above components, with formal syntax and security as defined in Section 2, our graded encoding scheme Γ consists of the algorithms detailed in the sections that follow. (See the introduction for an intuition.)

5.1 Setup

The **Setup** algorithm of Γ gets as input 1^λ and 1^κ. It samples parameters $pp_{\mathbb{G}} \leftarrow_{\$} \mathbf{Setup}_{\mathbb{G}}(1^\lambda)$ with $pp_{\mathbb{G}} := (\mathbb{G}, g, p, 1)$, generates two encryption key pairs $(pk_j, sk_j) \leftarrow_{\$} \mathbf{Gen}(1^\lambda)$ 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

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

a vector of κ elements in the base group $\mathbb{G}$, with κ the number of desired levels It then samples $lk \leftarrow_{\$} \mathbf{Gen_L}(1^\lambda)$, and sets

$$gpk := (pp_{\mathbb{G}}, pk_1, pk_2, [\boldsymbol{\omega}], lk) \ .$$

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 continues by generating a *binding* CRS $(crs', td_{ext}) \leftarrow_{\$} \mathbf{BCRS}(gpk)$, and also a *no-instance* of $\mathcal{L}_{lk}$ via $y \leftarrow_{\$} \mathbf{NoSam_L}(lk)$. It sets $crs := (crs', y)$. (The relation $\mathbf{R}$ that the NIZK should support will be defined shortly in Section 5.2.)

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 Sections 5.3 and 5.4, respectively. **Setup** also selects a seed hk for a randomness extractor and outputs the scheme parameters

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

5.2 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 in this paper. 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 5.3. 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.[7]

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 $\mathbf{Enc}(\cdot, pk_1)$ and $\mathbf{Enc}(\cdot, pk_2)$, respectively,[8] and π is a verifying NIZK proof under crs' that:

[7]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.

[8]This "honest-ciphertext-generation" condition is necessary for the (bi)linearity of our addition and multiplication algorithms. Unfortunately, this also prevents the sets $S_\ell^{(z)}$ from being efficiently recognizable.

(1) either c_1 and c_2 contain polynomials P_1 and P_2 of degree at most ℓ, 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 $(P_1, P_2, r_1, r_2, sk_1, sk_2)$ 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 $\leq \ell$ (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:

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

Note that there are two types of witnesses that can be used in proof generation for $\mathbf{R}_1$, namely (P_1, P_2, r_1, r_2) and (sk_1, sk_2).

Let $\mathbf{R_L}$ be the relation for the trapdoor language Λ. Relation $\mathbf{R}_2$, given gpk, an encoding, and a witness w_y, accepts iff $\mathbf{R_L}(lk, y, w_y)$ 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 5.1.

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 $\mathbf{Val}_\ell(h)$ 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 $\mathbf{Enc}(P_i, pk_i)$; this implies that these ciphertexts behave as expected under the homomorphic evaluation algorithm $\mathbf{Eval}$.) 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 $\mathbf{Enc}$. 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 $\mathbf{Eq}_\ell$ 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 $\mathbf{Eq}_\ell$ is only required to work with respect to consistent encodings.)

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.

5.3 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)).

```
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 $\neg(\mathbf{Val}_\ell(h) \wedge \mathbf{Val}_\ell(h'))$ 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. $P_1 \leftarrow \mathbf{Dec}(c_1, sk_1)$; $P_2 \leftarrow \mathbf{Dec}(c_2, sk_2)$
   $P_1' \leftarrow \mathbf{Dec}(c_1', sk_1)$; $P_2' \leftarrow \mathbf{Dec}(c_2', sk_2)$
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. $w_y' \leftarrow_{\$} \mathbf{WExt}(td_{ext}, ([z], c_1, c_2), \pi)$
   5.2. if $\neg\mathbf{R}_2(gpk, ([z], c_1, c_2, \ell), w_y')$ then return $\perp$
   5.3. $\pi'' \leftarrow_{\$} \mathbf{Prove}(gpk, crs, ([z''], c_1'', c_2''), w_y')$
6. else $\pi'' \leftarrow_{\$} \mathbf{Prove}(gpk, crs, ([z''], c_1'', c_2''), (sk_1, sk_2))$
7. return $([z''], c_1'', c_2'', \pi'', \ell)$
```

Figure 2: 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**.

Figure 2 details the operation of the addition circuit $\mathrm{C}_{\mathrm{Add}}$. A **PIO** of this circuit will be made public via the parameters 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 $\mathbf{Add}_\ell$ 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

```
CIRCUIT C_Mult[$gpk, crs, \omega, sk_1, sk_2, td_{ext}$]($\ell, \ell', h, h'$):        // for $1 \le \ell, \ell' \le \kappa - 1$
1. if $\neg(\mathbf{Val}_\ell(h) \wedge \mathbf{Val}_{\ell'}(h')) \vee \ell + \ell' > \kappa$ 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. $P_1 \leftarrow \mathbf{Dec}(c_1, sk_1)$; $P_2 \leftarrow \mathbf{Dec}(c_2, sk_2)$
   $P_1' \leftarrow \mathbf{Dec}(c_1', sk_1)$; $P_2' \leftarrow \mathbf{Dec}(c_2', sk_2)$
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. $w_y' \leftarrow_\$ \mathbf{WExt}(td_{ext}, ([z], c_1, c_2), \pi)$
   6.2. if $\neg\mathbf{R}_2(gpk, ([z], c_1, c_2), w_y')$ then return $\perp$
   6.3. $\pi'' \leftarrow_\$ \mathbf{Prove}(gpk, crs, ([z''], c_1'', c_2''), w_y')$
7. else $\pi'' \leftarrow_\$ \mathbf{Prove}(gpk, crs, ([z''], c_1'', c_2''), (sk_1, sk_2))$
8. If $(\ell + \ell' = \kappa)$ then return $[z'']$ else return $([z''], c_1'', c_2'', \pi'', \ell + \ell')$
```

Figure 3: 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**.

longer be established. (See also remark on validity on page 15.) Note that full compactness ensures that the ciphertexts output by $\mathbf{Add}_\ell$ are in the range of encryption, and hence they can be further operated on with **Eval**.

5.4 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 ℓ and ℓ' 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 $[zz'] = [(P * P')(\omega)]$.[9] Circuit $\mathrm{C}_{\mathrm{Mult}}$ is shown in Figure 3. 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 [AFH$^+$16] 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 $\ell < \kappa$.[10] 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.

[9]Observe that with the explicit knowledge of $P * P'$ and the powers $([\omega^i])_{1 \le i \le \kappa}$ it is also possible to compute $[zz']$ as long as $P * P'$ is of degree $\le \kappa$; this will be exploited in the security analysis in Section 7.

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

```
$\underline{\kappa\text{-Switch}_\Gamma^{\mathcal{A}}(\lambda)}$:
  $(pp;\omega) \leftarrow\!\!\$\ \mathbf{Setup}(1^\lambda, 1^\kappa)$        // $\omega$ generated within **Setup**
  $((P_{0,1}, P_{0,2}), (P_{1,1}, P_{1,2}), \ell, st) \leftarrow\!\!\$\ \mathcal{A}_1(pp, \omega)$
  $b \leftarrow\!\!\$\ \{0,1\};\ r_1, r_2 \leftarrow\!\!\$\ \{0,1\}^{\mathrm{rl}(\lambda)}$
  $c_1 \leftarrow \mathbf{Enc}(P_{b,1}, pk_1; r_1);\ c_2 \leftarrow \mathbf{Enc}(P_{b,2}, pk_2; r_2)$
  $\pi \leftarrow\!\!\$\ \mathbf{Prove}(gpk, crs, ([P_{b,1}(\omega)], c_1, c_2, \ell), (P_{b,1}, P_{b,2}, r_1, r_2))$
  $h_b \leftarrow ([P_{b,1}(\omega)], c_1, c_2, \pi, \ell)$
  $b' \leftarrow\!\!\$\ \mathcal{A}_2(h_b, st)$
  Return $(b = b')$
```

Figure 4: Game formalizing the indistinguishability of encodings. (This game is specific to our construction Γ from Section 5.) 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 ℓ. 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.

5.5 Sampling

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

$$\begin{aligned} h \leftarrow \big([z], c_1 = \mathbf{Enc}(P_1, pk_1; r_1), c_2 = \mathbf{Enc}(P_2, pk_2; r_2),\\ \pi = \mathbf{Prove}(gpk, crs, ([z]_i, c_1, c_2, \ell), (P_1, P_2, r_1, r_2); r), \ell\big)\ . \end{aligned} \tag{2}$$

Hence, our sampling algorithm $\mathbf{Sam}_\ell(z)$ sets $P_1(X) = P_2(X) = z \in \mathbb{Z}_p$ and computes an encoding through (2). We call these the *canonical* encodings of z, independently of ℓ. We note that this procedure is that in [AFH$^+$16] adapted to the generalized notion of polynomial representations.

5.6 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 $[z]$.

6 Indistinguishability of Encodings

We show that a key property used by AFHLP in the analysis of their multilinear map [AFH$^+$16, 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 4. 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 6.1 (Encoding switch)**.** *Let Γ be the GES constructed in Section 5 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 $\lambda \in \mathbb{N}$*

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

The proof of this result follows largely that in [AFH$^+$16] and we include it for completeness in Appendix B. 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 5 shows the steps used in the proof of the theorem. We use helper Lemma 6.2 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. See Appendix B for the full proof.

Gm.	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	$\boxed{\text{hiding}}$	$\boxed{\in \mathcal{L}_{lk}}$	$\boxed{w_y}$	$\boxed{\underline{sk_1}, w_y}$	$P_{b,1}$	$P_{b,2}$	Lemma 6.2 ($i = 1$)
2	hiding	$\in \mathcal{L}_{lk}$	w_y	$\underline{sk_1}, w_y$	$P_{b,1}$	$\boxed{P_{1,2}}$	IND-CPA wrt. pk_2
3	$\boxed{\text{binding}}$	$\boxed{\notin \mathcal{L}_{lk}}$	$\boxed{sk_1, sk_2, td_{ext}}$	$\boxed{\underline{sk_1}, sk_2, td_{ext}}$	$P_{b,1}$	$P_{1,2}$	Lemma 6.2 (reverse, $i = 1$)
4	$\boxed{\text{hiding}}$	$\boxed{\in \mathcal{L}_{lk}}$	$\boxed{w_y}$	$\boxed{\underline{sk_2}, w_y}$	$P_{b,1}$	$P_{1,2}$	Lemma 6.2 ($i = 2$)
5	hiding	$\in \mathcal{L}_{lk}$	w_y	$\underline{sk_2}, w_y$	$\boxed{P_{1,1}}$	$P_{1,2}$	IND-CPA wrt. pk_1
							Encoding indep. of b

Figure 5: Outline of the proof steps of Theorem 6.1. 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 with a binding crs' and $y \notin \mathcal{L}_{lk}$. The addition and multiplication circuits are defined in Figures 2 and 3, respectively.

Game$_1$: We change the public parameters so that they include a *hiding* crs', a yes instance y via $\mathbf{YesSam_L}(lk)$ and obfuscations of circuits $\widehat{\mathrm{C}}_{\mathrm{Add}}$ and $\widehat{\mathrm{C}}^{(1)}_{\mathrm{Mult}}$ (see Figure 6). 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. The difference with the previous game can be bounded by our helper Lemma 6.2 with $i = 1$, where we rely on PIO security, CRS indistinguishability, and the membership problem.

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$_1$.

Game$_3$: The public parameters are changed back to include a binding crs', a no-instance $y \notin \mathcal{L}_{lk}$ and a (PIO) obfuscation of the original circuits $\mathrm{C}_{\mathrm{Add}}$, $\mathrm{C}_{\mathrm{Mult}}$ with both decryption keys hardwired. The difference with the previous game is bounded again via Lemma 6.2 (in the reverse direction and with $i = 1$).

Game$_4$: This transitions is defined analogously to that introduced in Game$_1$ except that this time we invoke Lemma 6.2 with $i = 2$ and switch to circuits $\widehat{\mathrm{C}}_{\mathrm{Add}}$ and $\widehat{\mathrm{C}}^{(2)}_{\mathrm{Mult}}$. Observe that knowledge of sk_1 is no longer needed.

Game$_5$: This transitions is defined analogously to that introduced in Game$_2$. The only difference is that this game generates the *first* challenge ciphertext c_1 by encrypting $P_{1,1}$ even when $b = 0$.

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 0. □

In the proof of Theorem 6.1, we need the next Lemma for changing the addition and multiplication circuits to "forget" (one or both) the secret keys and the extraction trapdoor. The proof can be found in Appendix C.

```
CIRCUIT $\widehat{\mathrm{C}}_{\mathrm{Add}}[gpk, crs, w_y](\ell, h, h')$:
1. if $\neg(\mathbf{Val}_\ell(h) \wedge \mathbf{Val}_\ell(h'))$ 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. $\pi'' \leftarrow_\$ \mathbf{Prove}(gpk, crs, ([z''], c_1'', c_2'', \ell), w_y)$
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 $\neg(\mathbf{Val}_\ell(h) \wedge \mathbf{Val}_{\ell'}(h')) \vee \ell + \ell' > \kappa$ 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. $P_i \leftarrow \mathbf{Dec}(c_i, sk_i)$; $P_i' \leftarrow \mathbf{Dec}(c_i', sk_i)$     // depends on $sk_i$ only
5. $z'' \leftarrow (P_i * P_i')(\omega)$
6. $\pi'' \leftarrow_\$ \mathbf{Prove}(gpk, crs, ([z''], c_1'', c_2'', \ell + \ell'), w_y)$
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 6: **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.

Lemma 6.2 (Forgetting secret keys)**.** *Let Γ be the GES from Section 5 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* $\mathbf{Setup}^{(i)}$ *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 6. Let*

$$\mathbf{Adv}^{\kappa\text{-forget}}_{\Gamma,i,\mathcal{A}}(\lambda) := 2 \cdot \Pr\big[pp_0 \leftarrow_\$ \mathbf{Setup}(1^\lambda, 1^\kappa);\ pp_1 \leftarrow_\$ \mathbf{Setup}^{(i)}(1^\lambda, 1^\kappa);\ b \leftarrow_\$ \{0,1\}; b' \leftarrow_\$ \mathcal{A}(pp_b)\ :\ b = b'\big] - 1\ .$$

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 $\lambda \in \mathbb{N}$*

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

7 Hardness of MDDH

We are now ready to show that MDDH is hard for our GES. We improve [AFH+16] by providing a simpler and tighter proof of security. One corollary of our result is that there are no "zeroizing" attacks on our scheme as such attacks immediately lead to the break of MDDH [CHL+15, CGH+15, GGH13a]. We start by providing formal definition of MDDH as well as the strong DDH problem whose hardness we assume in our analyses.

THE q-SDDH PROBLEM [BB04, ZSS04]. For $q \in \mathbb{N}$ we say that the q-SDDH problem is hard for a group $\mathbb{G}$ if

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

where game q-SDDH$^{\mathcal{A}}_{\mathbb{G}}(\lambda)$ is shown in Figure 7 (left). We note that this assumption can only hold in *asymmetric* pairing-friendly groups. (With such asymmetric pairings, we could then implement, e.g., the

```
q-SDDH_G^A(λ)
  $pp_{\mathbb{G}} \leftarrow_\$ \mathbf{Setup}_{\mathbb{G}}(1^\lambda)$
  $b \leftarrow_\$ \{0,1\}$
  $\omega, \tau_0 \leftarrow_\$ \mathbb{Z}_p$
  $\tau_1 \leftarrow \omega^{q+1} \pmod p$
  $b' \leftarrow_\$ \mathcal{A}(pp_{\mathbb{G}}, \{[\omega^i]\}_{i=1}^q, [\tau_b])$
  Return $(b = b')$

κ-MDDH_Γ^A(λ)
  $pp \leftarrow_\$ \mathbf{Setup}(1^\lambda, 1^\kappa)$
  $b \leftarrow_\$ \{0,1\}$
  $a_1, \dots, a_{\kappa+1}, z \leftarrow_\$ \mathbb{Z}_p$
  $h_i \leftarrow_\$ \mathbf{Sam}_1(a_i)$
  $h_0^* \leftarrow_\$ \mathbf{Sam}_\kappa(z)$
  $h_1^* \leftarrow \mathbf{Mult}(h_1, \dots, h_\kappa)^{a_{\kappa+1}}$
  $b' \leftarrow_\$ \mathcal{A}(pp, \{h_i\}_{i=1}^{\kappa+1}, h_b^*)$
  Return $(b = b')$
```

Figure 7: **Left**: The SDDH problem. Here $p = p(\lambda)$ denotes the group order implicit in pp. **Right**: The MDDH problem. 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.

dual-mode NIZK proof system from [GS08].) It is not too difficult to show via re-randomization of the group generator that hardness of q-SDDH implies that of $(q-1)$-SDDH. We use this fact to simplify our theorem statement below.

The κ-MDDH problem [BS03, GGH13a]. For $\kappa \in \mathbb{N}$ we say that the κ-MDDH problem is hard for a GES Γ if

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

where game κ-MDDH$_\Gamma^{\mathcal{A}}(\lambda)$ is shown in Figure 7 (middle).

The $(\kappa, m, n, r_0, r_1, l)$-RANK problem [EHK+13]. 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 Γ if

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

where game $(\kappa, m, n, r_0, r_1, l)$-RANK$_\Gamma^{\mathcal{A}}(\lambda)$ is shown in Figure 7 (right).

7.1 Hardness of MDDH

Recall that the GES of Section 5 represents an element $z \in \mathbb{Z}_p$ at level ℓ with polynomials P_1 and P_2 of degree at most ℓ such that $P_j(\omega) = z$.

Theorem 7.1 (κ-SDDH $\implies$ κ-MDDH)**.** *Let Γ be the GES constructed in Section 5 with respect to a base group $\mathbb{G}$ and an X-IND-secure probabilistic obfuscator* **PIO**.

Then, assuming the κ-SDDH assumption (see Fig. 7) holds in $\mathbb{G}$, and using our switching lemma, the κ-MDDH assumption holds in Γ.

More specifically, 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 $\lambda \in \mathbb{N}$*

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

Outline. We provide a simpler proof compared to that of [AFH+16, Theorem 6.2] at the expense of relying on the slightly stronger κ-SDDH (instead of the $(\kappa-1)$-SDDH) problem. 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 [AFH+16, 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 outline the proof along a sequence of $\kappa + 5$ games here and leave the full details to Appendix D.

Game$_0$**:** This is the κ-MDDH problem (Figure 7, 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 $\mathbf{Sam}_1(a_i)$.

Game$_1$–Game$_{\kappa+1}$**:** In these games we gradually switch the polynomials representations for level-1 encodings h_i for $1 \leq i \leq \kappa+1$ so that they take the form

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

These polynomials are still valid and their degrees are *exactly* 1. Hence when multiplied together, the resulting polynomial will be of degree $s(\kappa+1)$. Each of these hops can be bounded via the κ-Switch game via Theorem 6.1.

Game$_{\kappa+2}$**:** This game only introduces a conceptual change: a_i for $1 \leq i \leq \kappa+1$ are generated as $a_i+\omega$. The distributions of these values are still uniform and the exponent of the MDDH challenge when $b=1$ is now

$$z_1 = \prod_{i=1}^{\kappa+1} (a_i + \omega) \ ,$$

which is a polynomial in ω of degree κ.

Game$_{\kappa+3}$**:** In this game we replace $\mathrm{C}_{\mathrm{Mult}}$ with $\mathrm{C}^*_{\mathrm{Mult}}$, a circuit that uses the implicit values $[\omega^i]$ for $0 \leq i \leq \kappa$ in steps 5 and 6. (Note that $[P(\omega)]$ can be computed using $[\omega^i]$ when the coefficients of P are explicitly known.) This change does not affect the functionality of the multiplication circuit and hence we can bound this hope via PIO security. As a result, the explicit knowledge ω is no longer needed to generate the multiplication circuit.

Game$_{\kappa+4}$**:** In this game, we replace $[\omega^\kappa]$ with a random value $[\sigma]$ in challenge preparation. (Note that level-κ encodings correspond to the base group.) We can bound this hop via the κ-SDDH game.

In the final game the challenge exponent (when $b=1$) is fully randomized. This means that the challenge is independent of b in Game$_{\kappa+4}$, which concludes the proof. □

7.2 Downgrading attacks

It might appear that our GES could be subject to a "downgrading" attack as follow. 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 5.5).

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.

Acknowledgments

We thank the anonymous reviewers for their helpful comments, and Kenny Paterson and Geoffroy Couteau for useful discussions. In particular, we thank Shuai Han for pointing out several flaws as well as fixes in an earlier version of the paper.

References

[AFH$^+$16] Martin R. Albrecht, Pooya Farshim, Dennis Hofheinz, Enrique Larraia, and Kenneth G. Paterson. Multilinear maps from obfuscation. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A, Part I*, volume 9562 of *LNCS*, pages 446–473. Springer, Heidelberg, January 2016.

[AFP16] Hamza Abusalah, Georg Fuchsbauer, and Krzysztof Pietrzak. Constrained PRFs for unbounded inputs. In Kazue Sako, editor, *CT-RSA 2016*, volume 9610 of *LNCS*, pages 413–428. Springer, Heidelberg, February / March 2016.

[AS17] Prabhanjan Ananth and Amit Sahai. Projective arithmetic functional encryption and indistinguishability obfuscation from degree-5 multilinear maps. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part I*, volume 10210 of *Lecture Notes in Computer Science*, pages 152–181, 2017.

[BB04] Dan Boneh and Xavier Boyen. Short signatures without random oracles. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 56–73. Springer, Heidelberg, May 2004.

[BLR$^+$15] 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 [OF15], pages 563–594.

[BS03] Dan Boneh and Alice Silverberg. Applications of multilinear forms to cryptography. *Contemporary Mathematics*, 324:71–90, 2003.

[BW13] 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.

[BWZ14] Dan Boneh, Brent Waters, and Mark Zhandry. Low overhead broadcast encryption from multilinear maps. In Garay and Gennaro [GG14], pages 206–223.

[CG13a] Ran Canetti and Juan A. Garay, editors. *CRYPTO 2013, Part I*, volume 8042 of *LNCS*. Springer, Heidelberg, August 2013.

[CG13b] Ran Canetti and Juan A. Garay, editors. *CRYPTO 2013, Part II*, volume 8043 of *LNCS*. Springer, Heidelberg, August 2013.

[CGH$^+$15] 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 [GR15], pages 247–266.

[CHL$^+$15] 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.

[CLLT16] Jean-Sébastien Coron, Moon Sung Lee, Tancrède Lepoint, and Mehdi Tibouchi. Cryptanalysis of GGH15 multilinear maps. In Robshaw and Katz [RK16], pages 607–628.

[CLT13] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. Practical multilinear maps over the integers. In Canetti and Garay [CG13a], pages 476–493.

[CLT15] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. New multilinear maps over the integers. In Gennaro and Robshaw [GR15], pages 267–286.

[CLTV15] Ran Canetti, Huijia Lin, Stefano Tessaro, and Vinod Vaikuntanathan. Obfuscation of probabilistic circuits and applications. In Dodis and Nielsen [DN15], pages 468–497.

[DN15] Yevgeniy Dodis and Jesper Buus Nielsen, editors. *TCC 2015, Part II*, volume 9015 of *LNCS*. Springer, Heidelberg, March 2015.

[EHK$^+$13] Alex Escala, Gottfried Herold, Eike Kiltz, Carla Ràfols, and Jorge Villar. An algebraic framework for Diffie-Hellman assumptions. In Canetti and Garay [CG13b], pages 129–147.

[FHPS13] Eduarda S. V. Freire, Dennis Hofheinz, Kenneth G. Paterson, and Christoph Striecks. Programmable hash functions in the multilinear setting. In Canetti and Garay [CG13a], pages 513–530.

[GG14] Juan A. Garay and Rosario Gennaro, editors. *CRYPTO 2014, Part I*, volume 8616 of *LNCS*. Springer, Heidelberg, August 2014.

[GGH13a] 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.

[GGH$^+$13b] 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.

[GGH$^+$13c] Sanjam Garg, Craig Gentry, Shai Halevi, Amit Sahai, and Brent Waters. Attribute-based encryption for circuits from multilinear maps. In Canetti and Garay [CG13b], pages 479–499.

[GGH15] Craig Gentry, Sergey Gorbunov, and Shai Halevi. Graph-induced multilinear maps from lattices. In Dodis and Nielsen [DN15], pages 498–527.

[GGSW13] Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. Witness encryption and its applications. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 467–476. ACM Press, June 2013.

[GMS16] Sanjam Garg, Pratyay Mukherjee, and Akshayaram Srinivasan. Obfuscation without the vulnerabilities of multilinear maps. Cryptology ePrint Archive, Report 2016/390, 2016.

[GR07] Shafi Goldwasser and Guy N. Rothblum. On best-possible obfuscation. In Salil P. Vadhan, editor, *TCC 2007*, volume 4392 of *LNCS*, pages 194–213. Springer, Heidelberg, February 2007.

[GR15] Rosario Gennaro and Matthew J. B. Robshaw, editors. *CRYPTO 2015, Part I*, volume 9215 of *LNCS*. Springer, Heidelberg, August 2015.

[GS08] Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 415–432. Springer, Heidelberg, April 2008.

[GS12] Jens Groth and Amit Sahai. Efficient noninteractive proof systems for bilinear groups. *SIAM J. Comput.*, 41(5):1193–1232, 2012.

[HK08] Dennis Hofheinz and Eike Kiltz. Programmable hash functions and their applications. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 21–38. Springer, Heidelberg, August 2008.

[HSW13] Susan Hohenberger, Amit Sahai, and Brent Waters. Full domain hash from (leveled) multilinear maps and identity-based aggregate signatures. In Canetti and Garay [CG13a], pages 494–512.

[Lin16] Huijia Lin. Indistinguishability obfuscation from DDH on 5-linear maps and locality-5 prgs. Cryptology ePrint Archive, Report 2016/1096, 2016.

[LSS14] Adeline Langlois, Damien Stehlé, and Ron Steinfeld. GGHLite: More efficient multilinear maps from ideal lattices. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 239–256. Springer, Heidelberg, May 2014.

[LT17] Huijia Lin and Stefano Tessaro. Indistinguishability obfuscation from bilinear maps and block-wise local prgs. Cryptology ePrint Archive, Report 2017/250, 2017.

[MSZ16] Eric Miles, Amit Sahai, and Mark Zhandry. Annihilation attacks for multilinear maps: Cryptanalysis of indistinguishability obfuscation over GGH13. In Robshaw and Katz [RK16], pages 629–658.

[OF15] Elisabeth Oswald and Marc Fischlin, editors. *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*. Springer, Heidelberg, April 2015.

[PS15] Omer Paneth and Amit Sahai. On the equivalence of obfuscation and multilinear maps. Cryptology ePrint Archive, Report 2015/791, 2015.

[PST14] Rafael Pass, Karn Seth, and Sidharth Telang. Indistinguishability obfuscation from semantically-secure multilinear encodings. In Garay and Gennaro [GG14], pages 500–517.

[RK16] Matthew Robshaw and Jonathan Katz, editors. *CRYPTO 2016, Part II*, volume 9815 of *LNCS*. Springer, Heidelberg, August 2016.

[SW14] 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.

[ZSS04] 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.

A Details of the AFHLP Symmetric Multilinear Map

AFHLP [AFH+16] construct a symmetric κ-linear group scheme Γ relying on the following building blocks:

1. An algorithm $\mathbf{Setup}_{\mathbb{G}}$ 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 $\mathbf{Obf}$.
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 section that follow.

A.1 Setup

The algorithm **Setup** for Γ gets as input 1^λ and 1^κ. It samples parameters $pp_{\mathbb{G}} \leftarrow_\$ \mathbf{Setup}_{\mathbb{G}}(1^\lambda)$ with $pp_{\mathbb{G}} := (\mathbb{G}, g, p, 1)$, generates two encryption key pairs $(pk_j, sk_j) \leftarrow_\$ \mathbf{Gen}(1^\lambda)$ (for $j = 1, 2$), and a vector $\boldsymbol{\omega} \in \mathbb{Z}_p^\ell$ where $\ell \in \{2, 3\}$. $\mathbb{G}$ is called the *base group*. It then samples $lk \leftarrow_\$ \mathbf{Gen}_{\mathbf{L}}(1^\lambda)$, and sets

$$gpk := (pp_{\mathbb{G}}, pk_1, pk_2, [\boldsymbol{\omega}], lk) \ .$$

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

The setup algorithm continues by generating a common reference string $crs' \leftarrow_{\$} \mathbf{BCRS}(gpk)$ using the dual-mode NIZK procedure **BCRS**, and also a no-instance of $\mathcal{L}_{lk}$ via $y \leftarrow_{\$} \mathbf{NoSam_L}(lk)$. 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 Sections A.4 and A.5, respectively. **Setup** then outputs the scheme parameters

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

A.2 Encodings

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 addition of encodings (see Section A.4) 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 $\mathbf{Enc}(\cdot, pk_1)$ and $\mathbf{Enc}(\cdot, pk_2)$, 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 $(\mathbf{x}, \mathbf{y}, r_1, r_2, sk_1, sk_2)$ accepts iff $[z] \in \mathbb{G}$, the *representations* of $[z]$ as $\mathbf{x}, \mathbf{y} \in \mathbb{Z}_p^\ell$ are valid with respect to $[\boldsymbol{\omega}]$ in the sense that

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

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

$$\begin{gathered} c_1 = \mathbf{Enc}(\mathbf{x}, pk_1; r_1) \wedge c_2 = \mathbf{Enc}(\mathbf{x}, pk_2; r_2) \\ \vee \\ (pk_1, sk_1) = \mathbf{Gen}(sk_1) \wedge (pk_2, sk_2) = \mathbf{Gen}(sk_2) \wedge \mathbf{x} = \mathbf{Dec}(c_1, sk_1) \wedge \mathbf{y} = \mathbf{Dec}(c_2, sk_2) \end{gathered}$$

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 5.2 for more details.)

A.3 Equality

The equality algorithm $\mathbf{Eq}_1$ returns true iff their first components match in $\mathbb{G}$. The correctness follows from the fact that $\mathbb{G}$ has unique encodings.

A.4 Addition

This section gives a description of $\mathbf{Add}_1$ for adding level-1 encodings. The public parameters of the scheme contain an obfuscation of the circuit $\mathrm{C}_{\mathrm{Add}}$ shown in Figure 8 (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). $\mathbf{Add}_1$ 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 2 for the definitions.

```
Circuit  $\mathrm{C}_{\mathrm{Add}}[gpk, crs, sk_1, sk_2, td_{ext}; r](h, h')$:
1. if $\neg\mathbf{Val}_1(h) \vee \neg\mathbf{Val}_1(h')$ 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 $\mathbf{x} \leftarrow \mathbf{Dec}(c_1, sk_1); \mathbf{y} \leftarrow \mathbf{Dec}(c_2, sk_2)$
      $\mathbf{x}' \leftarrow \mathbf{Dec}(c_1', sk_1); \mathbf{y}' \leftarrow \mathbf{Dec}(c_2', sk_2)$
  4.2a if $([z] \neq [\langle \mathbf{x}, \boldsymbol{\omega} \rangle]) \vee ([z] \neq [\langle \mathbf{y}, \boldsymbol{\omega} \rangle])$ goto 5a
  4.2b else if $([z'] \neq [\langle \mathbf{x}', \boldsymbol{\omega} \rangle]) \vee ([z'] \neq [\langle \mathbf{y}', \boldsymbol{\omega} \rangle])$
       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 $w_y' \leftarrow \mathbf{WExt}(td_{ext}, ([z], c_1, c_2), \pi; r)$
  5a.2 if $\neg\mathbf{R}_2(gpk, (([z], c_1, c_2)), w_y')$ return $\perp$
  5a.3 $\pi'' \leftarrow \mathbf{Prove}(gpk, crs, ([z''], c_1'', c_2''), w_y'; r)$
5b. repeat 5a with $h'$    // $\mathbf{R}_1$ does not accept $h'$
5c. $\pi'' \leftarrow \mathbf{Prove}(gpk, crs, ([z''], c_1'', c_2''), (sk_1, sk_2); r)$
6. return $([z''], c_1'', c_2'', \pi'')$
```

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

Figure 8: **Top**: Circuit for addition of encodings. **Bottom**: Circuit implementing the multilinear map.

A.5 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 $\mathbf{x}_i$. It then uses the explicit knowledge of $\boldsymbol{\omega}$ to compute the output of the map as

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

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 $\mathbf{x}_i$ and $\boldsymbol{\omega}$ 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 8, bottom) will be made available through the public parameters and $\mathbf{e}$ is defined to run this circuit on its inputs.

A.6 Sampling

For sampling level-1 encodings, let $\mathbf{x}$ and $\mathbf{y}$ be vectors in $\mathbb{Z}_p^\ell$ satisfying $\langle \mathbf{x}, \boldsymbol{\omega} \rangle = \langle \mathbf{y}, \boldsymbol{\omega} \rangle$, set $[z] := [\langle \mathbf{y}, \boldsymbol{\omega} \rangle]$ (which can be computed using $[\boldsymbol{\omega}]$ and explicit knowledge of $\mathbf{x}$) and define the output of $\mathbf{Sam}_1$ to be

$$h \leftarrow \big([z], c_1 = \mathbf{Enc}(\mathbf{x}, pk_1; r_1), c_2 = \mathbf{Enc}(\mathbf{y}, pk_2; r_2),$$
$$\pi = \mathbf{Prove}(gpk, crs, ([z], c_1, c_2), (\mathbf{x}, \mathbf{y}, r_1, r_2); r) \big) .$$

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

A.7 Extraction

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

B Proof of Theorem 6.1: Indistinguishability of Encodings

Proof. We adapt the hybrids of [AFH$^+$16, Theorem 5.3] to the graded setting. In the last hybrid, the challenge encoding is drawn independently of the bit b, and therefore the advantage of any (even unbounded) adversary is zero. Below we let W_i denote the event that Game_i outputs 1.

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 5 shows the steps used in the proof of the theorem.

Game_0**:** This is the original κ-Switch game (see Figure 4).

Game_1**:** The public parameters are changed so that they include a hiding crs', a yes-instance $y \leftarrow_\$ \mathbf{YesSam_L}(lk)$ and (probabilistic) obfuscations of the circuits $\widehat{\mathrm{C}}^{(1)}_{\mathrm{Mult}}$, and $\widehat{\mathrm{C}}_{\mathrm{Add}}$ (see Figure 6). Recall that these circuits use the witness w_y to y to produce the output proofs π''. Therefore the *simultaneous* knowledge of decryption keys (sk_1, sk_2) is not needed. By Lemma 6.2 we have that

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

Game_2**:** As Game_1, but now 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 $c_1 \leftarrow \mathbf{Enc}(P_{b,1}, pk_1)$ for a random bit b, and $c_2 \leftarrow \mathbf{Enc}(P_{1,2}, pk_2)$. We claim that

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

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 $\mathcal{A}$ 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 $c_1 \leftarrow_\$ \mathbf{Enc}(P_{b,1}, pk_1)$ 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, $\pi \leftarrow_\$ \mathbf{Sim}(td_{zk}, x)$. 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 $y \leftarrow_\$ \mathbf{NoSam_L}(lk)$ and obfuscations of circuits $\mathrm{C}_{\mathrm{Add}}$ and $\mathrm{C}_{\mathrm{Mult}}$ of Figures 2 and 3. Once again by Lemma 6.2 we have that

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

Game_4**:** The public parameters are changed so that they include a hiding crs', a yes-instance $y \leftarrow_\$ \mathbf{YesSam_L}(lk)$ and obfuscations of circuits $\widehat{\mathrm{C}}^{(2)}_{\mathrm{Mult}}$ and $\widehat{\mathrm{C}}_{\mathrm{Add}}$ (see Figure 6). By Lemma 6.2

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

Game$_5$: 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 $c_1 \leftarrow \mathbf{Enc}(P_{1,1}, pk_1)$, and $c_2 \leftarrow \mathbf{Enc}(P_{1,2}, pk_2)$. 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 $\Pr[\mathrm{W}_5(\lambda)] = 1/2$ because the challenge encoding is generated using the same pair of polynomial representations $(P_{1,1}, P_{1,2})$ regardless of the value of the bit b. The proof of the theorem follows by collecting the terms above. □

C Proof of Lemma 6.2

Proof. We provide an outline of the game hops in Figure 9 and give the details next.

Gm.	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}	$\boxed{\underline{sk_i}, sk_{3-i}, td_{ext}}$	(sk_1, sk_2) or w'_y	**PIO**/soundness
2	binding	$\boxed{\in \mathcal{L}_{lk}}$	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	$\in \mathcal{L}_{lk}$	$\boxed{sk_1, sk_2, w_y}$	$\boxed{\underline{sk_i}, sk_{3-i}, w_y}$	(sk_1, sk_2) or w_y	**PIO**/unique w_y
4	$\boxed{\text{hiding}}$	$\in \mathcal{L}_{lk}$	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	$\in \mathcal{L}_{lk}$	$\boxed{w_y}$	$\boxed{\underline{sk_i}, w_y}$	$\boxed{w_y}$ (always)	**PIO**/WI

Figure 9: Outline of the proof of Lemma 6.2. 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$_0$: We start with a game that runs $\mathcal{A}$ on pp_0; that is with an obfuscation of $\mathrm{C}_{\mathrm{Add}}$ and $\mathrm{C}_{\mathrm{Mult}}$ (see Figures 2 and 3), 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$_5$**:** 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 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 $\widehat{\mathrm{C}}^{(i)}_{\mathrm{Mult}}$. □

D Proof of Theorem 7.1: The MDDH Problem

Proof. 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 game as shown in the middle of Figure 7.

Game$_1$–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

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

These representation polynomials are valid and of degree *exactly* 1, the maximum allowed degree at level 1. 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 game κ-Switch of Figure 4. 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 1). 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.

Game$_{\kappa+2}$**:** The i-th source exponent is changed to $a_i' = a_i + \omega$ 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 $P_{\kappa+2,1}(X) \equiv P_{\kappa+2,2}(X) = X + a_i$, which has the same degree as the polynomials in Game$_{\kappa+1}$. This means that the exponent of the target encoding h_b^* when $b = 1$ is

$$z_1^* = Q(\omega) := (\omega^s + a_1) \cdots (\omega^s + a_{\kappa+1}) \ . \tag{3}$$

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

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

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 $\big([1], [\omega], \ldots, [\omega^{\kappa+1}]\big)$, and the explicit knowledge of the coefficients $(q_0, \ldots, q_{\kappa+1})$ of polynomial $Q(X)$ obtained by expanding Equation 3. This change is purely conceptual.

The second difference is that we obfuscate circuit $\mathrm{C}^*_{\mathrm{Mult}}$ which has the powers $\big([1], [\omega], \ldots, [\omega^{\kappa}]\big)$ 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 $\big([1], [\omega], \ldots, [\omega^{\kappa}]\big)$. 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 ℓ (respectively, ℓ'), and therefore P'' is of degree at most $(\ell + \ell') \leq \kappa$. 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:

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

Game$_{\kappa+4}$: We regard the degree $\kappa+1$ polynomial $Q(X)$ of Equation (3) as a multivariate $\mathbb{Z}_p$-polynomial $Q'(Y_1, \ldots, Y_{\kappa+1})$ in $\kappa+1$ 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

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

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

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

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

To see that $\Pr[\mathrm{W}_{\kappa+4}] = 1/2$ it suffices to show that in 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 $f(X) := Q(\omega, \ldots, \omega^{\kappa}, X)$ defines a bijection over $\mathbb{Z}_p$ mapping a uniform τ into a uniform $z_1^* = f(\tau)$. $\square$