Stronger Multilinear Maps from Indistinguishability Obfuscation

Navid Alamati* Hart Montgomery † Sikhar Patranabis‡

May 26, 2020

Abstract

We show how to construct new multilinear maps from subexponentially secure indistinguishability obfuscation (iO) and (relatively) standard assumptions. In particular, we show how to construct multilinear maps with arbitrary predetermined degree of multilinearity where each of the following assumptions hold: SXDH, joint-SXDH, exponent-DDH and all other assumptions implied by them (including k-party-DDH, k-Lin and its variants). Our constructions almost identically achieve the full functionality of the "dream version" definition of multilinear maps as defined in the initial work of Garg *et al.* (Eurocrypt'13). Our work substantially extends a previous line of works including that of Albrecht *et al.* (TCC'16) and Farshim *et al.* (PKC'18), which showed how to build multilinear maps endowed with weaker assumptions (such as multilinear DDH and other related assumptions) from iO.

A number of recent works have shown how to build iO from multilinear maps endowed with hardness assumptions; one example would be the work of Lin and Tessaro (Crypto'17) which shows how to construct iO from subexponentially secure SXDH-hard multilinear maps and some (subexponentially secure) plausible assumptions. Coupled with any one of these constructions, our results here can be seen as formally proving the equivalence of iO and multilinear maps/graded encodings (modulo subexponential reductions and other relatively standard assumptions) for the first time.

*University of Michigan.
†Fujitsu Laboratories of America.
‡ETH Zürich.

1 Introduction

Indistinguishability obfuscation (iO) [BGI+01] is a powerful primitive that offers enormous potential in terms of cryptographic constructions. In fact, almost every cryptographic primitive can be built from iO (and some other mild assumptions). This includes many strong primitives such as functional encryption [GGH+13b], multi-party noninteractive key exchange [BZ14], and much more [SW14, GGHR14, HSW14, BP15]. Due to its many applications, iO has been given its own "complexity world" called obfustopia [GPSZ17].

Unfortunately, cryptographers have struggled to come up with secure candidate constructions for iO based on standard assumptions. Garg *et al.* [GGH+13b] proposed the first candidate construction of iO based on the construction of a graded encoding (a generalization of multilinear maps) due to Garg, Gentry, and Halevi [GGH13a]. This spurred a huge interest in building new multilinear maps/graded encodings, and several other candidate schemes [CLT13, GGH15, Zim15, AB15] were proposed subsequently.

However, many of these candidate constructions were cryptanalized, starting with the work of Cheon *et al.* [CHL+15] on the cryptanalysis of the multilinear map defined in [CLT13]. More attacks on multilinear maps and their implied iO constructions followed [MSZ16, HJ16, CLLT16], breaking all of the original multilinear map schemes. Subsequent attempts to "immunize" the existing constructions against these attacks [CLT15, BMSZ15, GMM+16] were also shown to be vulnerable [MSZ16, CGH17]. To our knowledge, there is only one currently published multilinear map that has not been attacked [MZ18], and it is has only recently been shown to imply iO [AMP20].[1]

The multitude of attacks against multilinear maps convinced the cryptographic community to focus on alternative ways of constructing iO. One such approach that received considerable attention was building iO based on functional encryption (FE). The authors of [AJ15, BV15] showed that subexponentially secure *single-key compact* FE (in conjunction with some other standard assumptions) implies iO. They also showed how to construct single-key compact FE from many-key FE. These works, together with some previous works [GGH+13b, Wat15] that showed how to build *many-key* FE from iO and standard assumptions, established an equivalence between (subexponentially secure) compact FE and iO (modulo certain plausible assumptions).

This naturally led to a proliferation of attempts to realize compact FE (and thus iO). A series of works [Lin16, LV16, LT17, AS17] showed how to realize compact FE based on *low-degree* multilinear maps and additional novel techniques such as local PRGs. In fact, in [LT17], the authors showed a construction of compact FE from bilinear maps and 2-blockwise local PRGs, which seemingly paved the way for a secure iO construction from standard assumptions. However, Lombardi and Vaikuntanathan [LV17] and Barak *et al.* [BBKK18] independently showed that 2-blockwise local PRGs could never be constructed securely, implying that such iO constructions cannot be securely instantiated.[2]

Most of the recent works on iO [Agr19, AM18, AJL+19, JLMS19] have continued down the compact FE road based on newer assumptions such as perturbation resilient generators.[3] A recent result of Brakerski *et al.* [BDGM20] shows how to build iO from a new primitive called *split FHE*. Unfortunately, many of these new assumptions are not yet well-understood. In fact, some of these recent results have already been cryptanalyzed [AP20].

1.1 Multilinear Maps and iO

The reported attacks on multilinear maps/graded encodings, together with the equivalence results between compact FE and iO, have meant that multilinear maps/graded encodings have received considerably less attention in recent years. A natural question to ask is: are multilinear maps (endowed with certain hardness assumptions), in fact, *stronger* than iO; in other words, is secure iO is *more likely* to exist than secure multilinear maps/graded encodings?

In this paper, we revisit this question. Based on existing cryptographic constructions and known attacks, the answer seems to be "yes," at least for multilinear maps that are powerful enough to imply iO. Roughly speaking, multilinear maps can be divided into two broad classes depending on the nature of the hardness assumption that they are endowed with: multilinear maps with hardness assumptions over the source groups, and multilinear maps with hardness assumptions over the target groups.

[1] Throughout this paper, we use multilinear maps and graded encodings interchangeably. Unless otherwise specified, multilinear maps refer to "graded" multilinear maps and not "one-shot" multilinear maps.

[2] More generally, [LT17] showed a construction of iO from k-linear maps and k-blockwise local PRGs (plus some standard assumptions). The scheme is not known to be broken for $k \geq 3$.

[3] We refer the reader to [HB15] for a survey of multilinear maps and iO.

"Source" and "Target" Group Assumptions. Suppose we consider an asymmetric multilinear map $e : \mathbb{G}_1 \times \ldots \times \mathbb{G}_\ell \rightarrow \mathbb{G}_T$ where each group is of order q. Examples of hardness assumptions over the "source group" include the SXDH assumption, which informally states that for each group $\mathbb{G}_i$, given some generator $g_i \in \mathbb{G}_i$, it is hard to distinguish between the tuples $\left(g_i, g_i^a, g_i^b, g_i^{ab}\right)$ and $\left(g_i, g_i^a, g_i^b, g_i^c\right)$, where $a, b, c \in \mathbb{Z}_q$ are chosen uniformly at random.[1]

Next, let's consider a symmetric multilinear map $e : \mathbb{G}^\ell \rightarrow \mathbb{G}_T$ where both $\mathbb{G}$ and $\mathbb{G}_T$ are of order q. An example of an assumption in the target group would be the multilinear DDH assumption, which informally states that, given a generator $g \in \mathbb{G}$ and $\ell + 1$ group elements $g^{a_1}, \ldots g^{a_\ell}, g^{a_{\ell+1}}$ which are encodings of random elements $a_1, \ldots, a_\ell, a_{\ell+1} \in \mathbb{Z}_q$, the term $e\left(g, \ldots, g\right)^{a_1 \ldots a_\ell a_{\ell+1}}$ is computationally indistinguishable from a uniformly random element in $\mathbb{G}_T$.

There are no known constructions of iO based on multilinear map endowed with only a target group assumption. This may be explained as follows: typically iO constructions based on multilinear maps involve encoding elements in the source groups. As a result, the security proofs explicitly require indistinguishability assumptions over the source groups. In other words, assumptions over the target groups are typically insufficient when arguing security of such iO constructions.

Building iO from Multilinear Maps. We already know how to build iO from many "source group" hardness assumptions over multilinear maps/graded encodings. Gentry *et al.* [GLSW15] showed how to build iO from a multilinear map where the multilinear subgroup decision assumption holds over the source group. Very recently, Alamati *et. al* [AMP20] generalized [GLSW15] to show that multilinear maps with minimal assumptions (such as SXDH or matrix-DDH [EHK+13]) imply iO.

Unfortunately, the above constructions need to assume subexponential hardness of the underlying multilinear maps in order to achieve iO, but this seems somewhat inherent and hard to avoid. We refer the reader to [GLSW15] for a detailed discussion on this issue.

Building Multilinear Maps from iO. On the other hand, only a few works have tried to build multilinear maps from iO. The first result in this direction was the construction of a self-bilinear map with auxiliary information [YYHK14]. While this enabled applications like multiparty noninteractive key exchange (NIKE), the authors did not consider decisional assumptions that could potentially imply iO.

Subsequently, the authors of [AFH+16] showed how to construct from iO (and some other reasonably standard assumptions) a "one-shot" (i.e., not graded) multilinear map where the multilinear DDH assumption holds. The authors of [FHHL18] further modified the construction in [AFH+16] to build a *graded* multilinear map where the MDDH assumption held. Unfortunately, we note that the multilinear DDH assumption is a "target group" assumption, and we do not know how to build iO from such assumptions.

So the state of the art is currently the following: we know how to build multilinear maps/graded encodings endowed with target group assumptions from iO and (relatively) standard assumptions. But we only know how to build iO from multilinear maps/graded encodings with certain source group assumptions. This apparently suggests that multilinear maps with such source group assumptions are plausibly strictly stronger than iO.

Further Difficulties and diO. There also seem to be many strong barriers for building multilinear maps with "source group" assumptions from iO. For instance, suppose that we want to build an asymmetric multilinear map endowed with the SXDH assumption. This would require us to publish circuits C_A and C_M that add and multiply encodings of elements. Assume for the moment that we only want to publish a "one-shot" zero-test circuit C_Z that computes whether or not a level-ℓ product is zero.

A natural approach might be to use obfuscation to try to hide secret information inside the circuits C_A, C_M and C_Z that would make it possible to process the encodings appropriately. However, we have the following difficulty: what if the adversary performs a sequence of addition and multiplication operations involving the SXDH challenge terms that results in a zero-encoding when the SXDH is "real", and a non-zero-encoding when the SXDH challenge is "random"? The zero-test circuit must behave differently on the resultant encoding in these two cases. Hence, standard iO is seemingly insufficient here since the obfuscated programs do not have identical outputs on known inputs.

[1] Note that SXDH can only be valid in an asymmetric multilinear map.

If we want to obfuscate programs that are not functionally equivalent on certain inputs, then we seemingly require the stronger notion of *differing inputs obfuscation* [ABG+13] (diO). Unfortunately, there are some impossibility results on diO for general circuits [BSW16, GGHW17], and only few instances of diO for certain restricted class of circuits are known to be secure based on regular iO [BCP14]. So any iO-based construction of a multilinear map scheme with a plausible source group assumption seemingly needs to bypass diO lower bounds, which appears nontrivial.

Other Work. There has been some other work that attempts to unify the notions of multilinear maps. In [PS15], Paneth and Sahai introduced the notion of *polynomial jigsaw puzzles*, which are an abstraction of multilinear maps. They show that iO is unconditionally equivalent to polynomial jigsaw puzzles. Unfortunately, in hindsight it is unclear whether polynomial jigsaw puzzles accurately model multilinear maps with source group assumptions.

For instance, the authors of [AS15] show a black-box separation of iO from collision-resistant hash functions (CRHFs). On the other hand, any multilinear map where SXDH holds implies a simple CRHF [Dam88] in one of the source groups. So any construction of multilinear maps from iO seemingly requires additional assumptions.

In this paper, we ask the following fundamental question, the answer to which would have many implications for future work on iO and related primitives:

Are multilinear maps with useful source group assumptions stronger than iO? Or are they (up to subexponential security reductions and modulo certain standard assumptions) equivalent primitives?

1.2 Our Contributions

In this paper we answer this question by showing, perhaps surprisingly, that subexponentially secure iO in conjunction with some other standard assumptions implies multilinear maps endowed with most of the well-known (prime order) source group assumptions. More precisely, suppose that the following cryptographic primitives exist:

- A *probabilistic iO* [CLTV15] scheme (implied by subexponentially secure standard iO and subexponentially secure puncturable PRFs in NC^1).
- Fully homomorphic encryption (FHE) with message space $\mathbb{Z}_q$ for some (large) prime q.
- A dual-mode, simulation-extractable non-interactive zero knowledge argument system (e.g. Groth-Sahai [GS08]).
- A language $\mathcal{L}$ for which the membership problem is hard and whose "yes" instances have unique witnesses (such a language is implied by any DDH-hard group).

Given these primitives and some additional assumptions (specified below), we show how to build the following multilinear maps/graded encodings:

- An *asymmetric* multilinear map that is SXDH-hard, assuming additionally the existence of any DDH-hard group $\mathbb{G}$ of prime order q.
- An *asymmetric* multilinear map that is joint-SXDH-hard, assuming additionally the existence of any DDH-hard group of prime order q.
- An *asymmetric* multilinear map that is exponent-DDH-hard, assuming additionally the existence of any exponent-DDH-hard group of prime order q.
- A *symmetric* multilinear map with degree of multilinearity ℓ that is $(\ell + 1)$-exponent-DDH-hard, assuming additionally the existence of any power-DDH-hard group of prime order q.

On the Ingredients. We note that all of the aforementioned ingredients for our constructions, with the exception of piO, may be considered reasonably standard cryptoprimitives. In addition, the variant of piO needed for our constructions can be built from any "regular" iO scheme and puncturable PRF in NC^1 with subexponential security using the tools from [CLTV15]. We also note that this set of assumptions is a subset of those required by [FHHL18].

Other Source Group Assumptions. The EDDH assumption implies a whole host of other source group assumptions, including k-party-DDH, Casc, SCasc, k-Lin, k-ILin, and, of course, multilinear DDH. Hence, our constructions immediately imply multilinear map schemes where these other assumptions hold as well. We refer the reader to [EHK+13] for the details of these assumptions and their inter-relationships.

Supported Features. Our multilinear maps almost identically the full "dream version" of features that are defined and discussed in [GGH13a] and thus should be usable in any application of multilinear maps. The only feature that our constructions do not achieve is deterministic encodings, which would otherwise enable "classic/ideal" multilinear map functionality.

1.3 Implications and Discussion

Our work has a number of interesting implications to iO and multilinear maps that we discuss below.

iO and Multilinear Maps. Coupled with existing work (e.g. [LT17] and the recent work of [AMP20]), our work shows that multilinear maps and iO are equivalent up to subexponential security reductions and modulo certain reasonably standard assumptions. This means that iO is tied as tightly to multilinear maps as it is to compact FE (building iO from compact FE also, to our knowledge, requires subexponential security).

This observation could be interpreted in one of two ways. On the one hand, it could be thought of as a positive development for multilinear maps with useful source group assumptions (since we can now build many of them from iO). On the other hand, it could be thought of as a negative development for iO (since candidate constructions of multilinear maps have mostly been broken).

Most importantly, however, we hope that our findings lead to more insights on constructing both multilinear maps/graded encodings and iO, whether it be of the form of new constructions or impossibility results.

iO and Homomorphic Primitives. Our result also has interesting implications with respect to the relationship between iO and a generic primitive endowed with algebraic structure, namely a *ring key-homomorphic weak PRF* (RKHwPRF). The authors of [AMP20] showed that a "slotted" RKHwPRF (a slight weakening of a "classic" RKHwPRF) implies input-activated iO, which in turn implies standard iO under certain subexponential security assumptions [GLSW15]. They also showed that any multilinear map endowed with simple source group assumptions (e.g., SXDH) implies a slotted RKHwPRF.

Coupled with these results, our findings in this paper establish that, modulo certain subexponential security assumptions, iO is equivalent to slotted RKHwPRFs. In other words, RKHwPRFs are, in some sense, *obfustopia-complete*.

Bootstrapping Multilinear Maps. An interesting aspect of our work is that it paves the way for "bootstrapping" low-degree multilinear maps into multilinear maps with arbitrarily large degree endowed with similar hardness assumptions (albeit under subexponential security assumptions). For instance, the authors of [LT17] showed that assuming 3-blockwise local PRGs and some other relatively standard primitives, SXDH-hard trilinear maps imply iO. Under subexponential security assumptions, our work would then allow such an SXDH-hard trilinear map to be "bootstrapped" into an SXDH-hard multilinear map of any (predetermined) degree via iO. In fact, we can "bootstrap" to multilinear maps endowed with potentially *stronger* assumptions such as joint-SXDH and 2-exponent-DDH. This provides yet another motivation for building low-degree multilinear maps from standard assumptions.

Open Questions. Our work gives rise to many interesting open problems. For example, it is not immediately clear as to how our techniques could be extended to build multilinear maps endowed with composite-order group assumptions (e.g., the multilinear subgroup hiding assumption as defined in [GLSW15]). This leaves open the question of whether or not iO implies multilinear maps endowed with such assumptions.

We also leave it open to investigate if iO implies multilinear maps that are endowed with useful hardness assumptions as well as *unbounded* degree of multilinearity. The self-bilinear map from [YYHK14] does not seem to support hardness assumptions of the nature considered in this work (we refer the reader to [YYHK14] for detailed discussions). A

plausible approach could be to try and build a "classic" RKHwPRF as defined in [AMP20] from iO (at the moment, iO is only known to imply a weaker "slotted" version of RKHwPRF). A classic RKHwPRF bears resemblance to a multilinear map/multilinear map with unbounded degree of multilinearity; however, our current techniques do not suffice to build it from iO.

1.4 Paper Outline

The remainder of the paper is arranged as follows. Section 2 presents an overview of our techniques. Section 3 presents preliminary background material. Section 4 presents our construction and security proof for the SXDH-hard and joint-SXDH hard asymmetric multilinear maps. Section 5 presents our construction and security proof for the exponent-DDH hard asymmetric multilinear map. Finally, Section 6 presents our construction and security proof for the exponent-DDH hard symmetric multilinear map.

2 Technical Overview

In this section we give an overview of our multilinear maps/graded encoding constructions and some of the key ideas that we use. The starting point of our construction is the works of [AFH+16] and [FHHL18]. Since [FHHL18] is a generalization of [AFH+16], we will typically refer to it when discussing the ideas present in both works.

We will use our construction of an asymmetric multilinear map where the SXDH assumption holds as a working example throughout most of this overview, as it is the simplest of our constructions. We assume some basic understanding of multilinear maps/graded encodings in order to be understood. The reader may refer Section 3 for some preliminary background on multilinear maps and other cryptographic primitives.

2.1 Construction Overview

We provide a high-level overview of our construction of SXDH-hard asymmetric multilinear map (MMap)/graded encoding from iO and other cryptographic assumptions.

Encoding Structure. Each encoding in our construction consists of two FHE ciphertexts that are encryptions of the underlyig plaintext element $\alpha \in \mathbb{Z}_q$ under different public-key/secret-key pairs (the double encryption is a necessary component of the proof as explained later), a description of the "level" ℓ of the encoding, and a NIZK proof π that the encoding has been computed correctly. We can express this as:

$$\mathsf{Encode}\left(\alpha\right) = \left(\mathsf{FHE.Enc}_{\mathsf{pk}_0}\left(\alpha\right), \mathsf{FHE.Enc}_{\mathsf{pk}_1}\left(\alpha\right), \ell, \pi\right).$$

It is important to note that, like previous work, our representation of α will not be unique. Depending on what assumption we want to prove, we will a "slotted" represent of α for our encodings. For our construction of SXDH-hard asymeetric MMap, we will encode α as a four-tuple of the form $(\alpha_0, \alpha_1, \alpha_2, \alpha_2)$ with the restriction that, for some fixed $a, b, c \in \mathbb{Z}_q$,

$$\alpha = \alpha_0 + a \cdot \alpha_1 + b \cdot \alpha_2 + c \cdot \alpha_3.$$

The reader may observe this representation is structurally similar to a DDH tuple. For our "real" construction of an SXDH-hard MMap, we will only use the α_0 component to encode elements. However, we will use the full slotted representation in certain hybrid arguments in the proof of SXDH.

Addition, Inversion and Multiplication. For addition and inversion of encodings at the same level, as well as for multiplying encodings at appropriate levels, we generate probabilistic-iO (piO) obfuscations of circuits that broadly adhere to the following strategy:

- Verify the proof π to make sure the encoding is constructed correctly (if not, abort).
- Use the FHE secret keys to decrypt the ciphertexts.
- Perform the desired operation and create a valid encoding of the output at the appropriate level.

Extraction and Zero-Test. For extraction, we generate we generate a piO obfuscation of a circuit that works as follows:

- Verify the proof π to make sure the encoding is constructed correctly (if not, abort).
- Use the FHE secret keys to decrypt the ciphertexts and recover α.
- Output g^{α} where $g \in \mathbb{G}$ is the generator of a DDH-hard group of order q

Given the aforementioned extraction circuit, zero-test follows trivially. Note that we can extract and zero-test encodings at *every* level – an essential feature of the "dream" version of MMaps defined in [GGH13a].

Comparison with Previous Works. We present a high-level comparison of our construction with those in [AFH+16] and [FHHL18]. The key difference between our construction and these previous works is in how we choose to encode a plaintext element $\alpha \in \mathbb{Z}_q$. In [AFH+16], the authors encode each plaintext element α as a linear function, which inherently limits their functionality to a "one-shot" MMap. The authors of [FHHL18] generalize this representation to univariate polynomials of fixed degree, which allows them to achieve a graded MMap construction.

Our encoding strategy here can be seen as a further generalization, where we allow a larger class of functions. While most of our encoding strategies could be modeled as multivariate polynomials, we opt for the aforementioned representation for ease of exposition. We would like to point out that this generalization plays a crucial rule in achieving MMap constructions with stronger (source group) assumptions, as compared to the previous works.

2.2 Proof Overview

We now present a high-level overview of the proof strategy for our SXDH-hard MMap construction. The proof can be broadly divided into three steps: (a) transforming the encodings in the SXDH challenge from one representation to another such that these are computationally indistinguishable, (b) embedding a DDH challenge (over the group $\mathbb{G}$) into the transformed encodings, and (c) switching the encodings back to the original representation. The idea behind step (b) is as follows: when the DDH instance over the group $\mathbb{G}$ is real (respectively, random), the transformed encodings constitute a real (respectively, random) SXDH instance over the MMap. We expand more on these steps below.

Indistinguishability of Encodings. The cornerstone of steps (a) and (c) in the proof strategy mentioned above is *indistinguishability of encodings*. Recall that in our SXDH-hard MMap construction, we encode α as a tuple of the form $(\alpha_0, \alpha_1, \alpha_2, \alpha_3)$ with the restriction that for some fixed $a, b, c \in \mathbb{Z}_q$,

$$\alpha = \alpha_0 + a \cdot \alpha_1 + b \cdot \alpha_2 + c \cdot \alpha_3.$$

Now consider two possible ways of representing α. In the first case, we encode α using only α_0, while the remaining slots are unused (this is actually what is done in the real construction). In the second case, we represent α in a manner that additionally uses one or more of the remaining slots. *Indistinguishability of encodings* requires that given an encoding of α, a computationally bounded adversary cannot infer which of the two representations was used. Below, we present a high-level explanation for why our construction achieves this security notion.

One of the key tools that we invoke in our argument here is probabilistic iO (piO) security [CLTV15]. Recall that piO is an extension of regular iO that allows obfuscating randomized circuits/programs. Informally, piO security states that it should be hard to distinguish between the obfuscations of any two circuits whose output distributions at each input are computationally indistinguishable (possibly in the presence of some auxiliary input).

Now observe that given an encoding of α, a possible source from which the adversary could try and infer information about the representation of α is the pair of FHE ciphertexts in the encoding. By standard CPA-security, the ciphertexts in isolation do not leak information about the representation; however, recall that the piO circuits for the various MMap operations are hardwired with the corresponding FHE decryption keys sk_0 and sk_1. Hence it is not immediately obvious how we can argue that the FHE ciphertexts hide the representation of α.

Key Switching. We use a *key-switching* trick as follows: for each operation $\mathsf{op} \in \{\mathsf{Add}, \mathsf{Mult}, \mathsf{Inv}, \mathsf{Ext}\}$, we define a pair of auxiliary circuits $\widehat{C}_{\mathsf{op},0}$ and $\widehat{C}_{\mathsf{op},1}$. The first circuit is hardwired with *only* the first decryption key sk_0, while the second circuit is hardwired with *only* the second decryption key sk_1. We use a hybrid argument where we switch from the original circuits for these operations (which embeds both sk_0 and sk_1) to one of the aforementioned auxiliary circuits (say, $\widehat{C}_{\mathsf{op},0}$). We use the simulation-extractability of the dual-mode NIZK to argue that the output distributions of the original and auxiliary circuits are computationally indistinguishable. This in turn allows us to invoke piO security.

At this point, we can invoke CPA security of FHE to switch the representation of α underlying the first FHE ciphertext in the encoding. Similarly, in a subsequent hybrid that uses $\widehat{C}_{\mathsf{op},1}$ instead of $\widehat{C}_{\mathsf{op},0}$, we switch the representation of α underlying the second FHE ciphertext as well.

One remaining issue is that in some intermediate hybrid(s), the two FHE ciphertexts use different representations of α. We invoke the perfect zero-knowledge property of the NIZK in the hiding mode in conjunction with a language $\mathcal{L}$ with hard membership, to argue that a computationally bounded adversary cannot distinguish such a "malformed" encoding from a "real" encoding (where the representation of α underlying both FHE ciphertexts is identical).

Embedding DDH Challenges. We now describe the idea behind step (b) of the proof strategy mentioned above, namely embedding a DDH challenge in the SXDH challenge encodings. Recall that the SXDH assumption requires that the following indistinguishability holds for any level:

$$\left(\mathsf{Encode}\left(\alpha_0\right), \mathsf{Encode}\left(\alpha_1\right), \mathsf{Encode}\left(\alpha_0 \cdot \alpha_1\right)\right) \overset{c}{\approx} \left(\mathsf{Encode}\left(\alpha_0\right), \mathsf{Encode}\left(\alpha_1\right), \mathsf{Encode}\left(\alpha^*\right)\right),$$

where $\alpha_0, \alpha_1, \alpha^* \leftarrow \mathbb{Z}_q$. Now, letting $a = \alpha_0, b = \alpha_1$ and $c = \alpha_0 \cdot \alpha_1$ or $c = \alpha^*$ depending on whether the challenge is real or random (where a, b and c are as discussed above), one can use any one of the following representations for the challenge encodings:

$$\left((a,0,0,0), (b,0,0,0), (c,0,0,0)\right) \quad \text{or} \quad \left((0,1,0,0), (0,0,1,0), (0,0,0,1)\right).$$

In addition, by indistinguishability of encodings, an SXDH adversary should not be able to infer which of the aforementioned representations is used for the challenge encodings.

Now, consider a hybrid in which the challenge encodings use the second representation, and suppose that we are given a DDH tuple over the group $\mathbb{G}$ which is either of the form $(g^{\alpha_0}, g^{\alpha_1}, g^{\alpha_0 \cdot \alpha_1})$ or of the form $(g^{\alpha_0}, g^{\alpha_1}, g^{\alpha^*})$. Suppose for the moment that we could *hypothetically* set $a = \alpha_0, b = \alpha_1$ and $c = \alpha_0 \cdot \alpha_1$ or $c = \alpha^*$ depending on whether the DDH challenge is real or random. Now observe the following: when the DDH challenge is real (respectively, random), the challenge encodings constitute a real (respectively, random) SXDH instance.

Oblique Embedding. The aforementioned embedding technique does not work directly since the secret exponents α_0, α_1 and α^* in the DDH challenge are not known the reduction. To get around this issue, we use an *oblique embedding* technique that only uses the group elements available as part of the DDH challenge.

Addition and Inversion. To begin with, observe that addition and inversion of encodings do not require the knowledge of the constants a, b and c. In particular, given an encoding of $x = (x_0, x_1, x_2, x_3)$ and $y = (y_0, y_1, y_2, y_3)$, the reduction can exploit the homomorphic properties of the encryption scheme to compute encodings of the following:

$$x + y = (x_0 + y_0, x_1 + y_1, x_2 + y_2, x_3 + y_3) \quad , \quad -x = (-x_0, -x_1, -x_2, -x_3).$$

Multiplication. Multiplication of encodings requires some more care. To begin with, observe that given an encoding of $x = (x_0, x_1, x_2, x_3)$ and an encoding of $y = (y_0, 0, 0, 0)$ (i.e., the encoding of y uses a representation as in the real scheme), the reduction can exploit the homomorphic properties of the encryption scheme to compute an encoding of the following product:

$$x \cdot y = (x_0 \cdot y_0, x_1 \cdot y_1, x_2 \cdot y_2, x_3 \cdot y_3).$$

This still does not require the knowledge of the constants a, b and c.

On the other hand, if the encoding of y also uses the second representation (i.e., all slots can be used to encode the plaintext element), then evaluating the cross-product terms would require knowledge of the constants. We address this issue as follows.

First, we use the simulation-extractability of the NIZK in the binding mode to enforce that any challenge encoding created by the adversary would have to follow the representation used in the real scheme (i.e., only the first slot is used to encode the plaintext element). At the same time, we use the zero-knowledge property of the NIZK in the hiding mode to allow *only* the SXDH challenge encodings created by the reduction to use the second representation (i.e., all slots can be used to encode the plaintext element). By mode indistinguishability of NIZK, this ensures that the reduction never has to multiply two encodings such that both of them use the second representation (note that here we exploit the fact that the MMap is asymmetric).

Extraction and Zero-Testing. Finally, the reduction handles extraction as follows: it embeds the DDH challenge inside the extraction circuit. Given an encoding of $x = (x_0, x_1, x_2, x_3)$, the extraction circuit outputs

$$g^* = g^{x_0 + a \cdot x_1 + b \cdot x_2 + c \cdot x_3} = g^{x_0} \cdot g^{a \cdot x_1} \cdot g^{b \cdot x_2} \cdot g^{c \cdot x_3},$$

where $a = \alpha_0, b = \alpha_1$ and $c = \alpha_0 \cdot \alpha_1$ or $c = \alpha^*$ depending on whether the DDH challenge is real or random. Finally, zero-testing follows trivially from extraction.

We would like to point out that the aforementioned oblique embedding strategy is the essential component that allows us to prove stronger assumptions over our MMap constructions as compared to the previous works [AFH+16, FHHL18].

2.3 Putting It All Together

We now outline how to put together the indistinguishability of encodings with the oblique group embedding strategy for the overall proof of SXDH. Informally, the proof proceeds through a sequence of hybrids as follows:

- The first hybrid resembles the real scheme. In particular, the SXDH adversary is provided with a set of challenge encodings that follow the representation used in the real scheme (i.e., only the first slot is used to encode the plaintext element).
- Next, via a sequence of hybrids, the SXDH challenge encodings are switched to follow the second representation discussed above (i.e., all slots are used to encode the plaintext element).
- In the next hybrid, the reduction obliquely embeds a real DDH challenge into the SXDH challenge encodings as discussed above.
- In the next hybrid, the embedded DDH challenge is switched from real to random. This implicitly switches the SXDH challenge from real to random.
- Finally, via a sequence of hybrids, the SXDH challenge encodings are switched back to the representation used in the real scheme (i.e., only the first slot is used to encode the plaintext element).

A key observation with respect to the aforementioned proof strategy is that the only step where we potentially change the input/output behavior of the obfuscated circuits is the step where we switch from the real DDH tuple to the random DDH tuple. Note that in this step, we *do not* change the circuits themselves, and hence we do not need to invoke any form of obfuscation security to argue indistinguishability. This is what allows us to circumvent the initially perceived requirement of different-inputs iO and to relax our requirement to only piO. We refer the reader to the main body of the paper for a more formal description of the hybrids and detailed proofs of computational indistinguishability.

2.4 Other Graded Encodings

The discussion above only focused on our construction of an asymmetric MMap/graded encoding where the SXDH assumption holds. Our other constructions of asymmetric MMaps (where the joint SXDH and the 2-exponent-DDH assumptions hold) fit in the same overall proof framework. In fact, this proof framework can be viewed as a generic tool that allows us to achieve asymmetric MMaps with most of the well-known (prime order) source group assumptions [EHK+13].

Note that a technical requirement in the proof framework outlined above is that the adversary is restricted from multiplying challenge encodings (or encodings derived from challenge encodings). This ensures that the reduction

is not required to handle cross-product terms when obliquely embedding a hard problem instance into the challenge encodings. This restriction no longer applies when we want to construct a symmetric MMap endowed with hardness assumptions. At a high level, we get around this issue by strengthening the hardness assumption on the group $\mathbb{G}$ used in the construction.

Recall that the $(\ell+1)$-EDDH assumption over a degree-ℓ symmetric MMap requires that the following indistinguishability holds for any level:

$$\left(\mathsf{Encode}\left(\alpha\right), \mathsf{Encode}(\alpha^{\ell+1})\right) \overset{c}{\approx} \left(\mathsf{Encode}\left(\alpha\right), \mathsf{Encode}\left(\alpha^{*}\right)\right),$$

where $\alpha, \alpha^{*} \leftarrow \mathbb{Z}_q$. Since the adversary can pair challenge encodings at the same level, we resort to obliquely embedding an instance of a hardness assumption that would allow the reduction to simulate all possible cross-product terms resulting from such pairings. More specifically, we assume that the reduction is provided with a challenge set of group elements of one of the following forms:

$$\left(g, g^{\alpha}, g^{\alpha^2}, \ldots, g^{\alpha^{(\ell+1)(\ell+2)/2}}\right) \quad \text{or} \quad \left(g, \left\{g^{\alpha^i \cdot (\alpha^*)^j}\right\}_{i+j\in[\ell]}\right),$$

where $g \leftarrow \mathbb{G}$ and $\alpha, \alpha^{*} \leftarrow \mathbb{Z}_q$. We refer to this indistinguishability assumption as the "strong"-$(\ell+1)$-EDDH assumption over the group $\mathbb{G}$. In the main body of the paper, we prove that this assumption is implied by the power-DDH assumption over the group $\mathbb{G}$.

3 Preliminaries

We recall the definition of a dual-mode NIZK proof system. We adopt the notation from [CKWZ13].

Definition 3.1. A tuple of three algorithms (G, P, V) is said to be noninteractive proof system for a language $L \in \mathsf{NP}$ if it satisfies the following properties:

- Completeness: For any $x \in L$ and any witness w for x we have

$$\Pr[\mathsf{crs} \leftarrow G(1^\lambda); \pi \leftarrow P(\mathsf{crs}, x, w) : V(\mathsf{crs}, x, \pi) = 1] = 1.$$

- Soundness: For any attacker $\mathcal{A}$, if $\mathsf{crs} \leftarrow G(1^\lambda)$ and $(x, \pi) \leftarrow \mathcal{A}(\mathsf{crs})$ then

$$\Pr[V(\mathsf{crs}, x, \pi) = 1 \wedge x \notin L] \leq \mathrm{negl}\,.$$

Definition 3.2. A noninteractive proof system (G, P, V) is said to be dual-mode NIZK if there are efficient simulators $\mathcal{S}$ and $\overline{\mathcal{S}}$ with the following properties:

- Indistinguishability of modes: If $\mathsf{crs}_0 \leftarrow G(1^\lambda)$ and $(\mathsf{crs}_1, \mathsf{st}) \leftarrow \mathcal{S}(1^\lambda)$ then $\mathsf{crs}_0 \overset{c}{\approx} \mathsf{crs}_1$.
- Simulation in ZK mode: For any $\mathcal{A}$ if $(\mathsf{crs}_1, \mathsf{st}) \leftarrow \mathcal{S}(1^\lambda)$ and $(x, w) \leftarrow \mathcal{A}(\mathsf{crs}, \mathsf{st})$ then

$$\Pr[\pi \leftarrow P(\mathsf{crs}, x, w) : \mathcal{A}(\pi) = 1] - \Pr[\pi \leftarrow \overline{\mathcal{S}}(\mathsf{st}, x) : \mathcal{A}(\pi) = 1] = 0.$$

Here we mention the definition of fully homomorphic encryption for bits. The following definition naturally generalizes to an arbitrary message space (say $\mathbb{Z}_q$). We remark that in the following definition we assume that the evaluation key is included as part of the public key.

Definition 3.3. A fully homomorphic encryption (FHE) scheme (for bits) is a tuple of four algorithms $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ such that the tuple $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is a CPA-secure PKE scheme and the evaluation algorithm Eval satisfies homomorphism and compactness properties as defined below:

- CPA security: If $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$ then for any messages m_0 and m_1 we have

$$\mathsf{Enc}(\mathsf{pk}, m_0) \overset{c}{\approx} \mathsf{Enc}(\mathsf{pk}, m_1).$$

- Homomorphism: For any (boolean) function $f : \{0,1\}^\ell \to \{0,1\}$ and any sequence of ℓ messages $m_1, \ldots, m_\ell$ if $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$ and $c_i \leftarrow \mathsf{Enc}(\mathsf{pk}, m_i)$ then

$$\Pr[\mathsf{Dec}(\mathsf{sk}, \mathsf{Eval}(\mathsf{pk}, c_1, \ldots c_\ell)) \neq f(m_1, \ldots, m_\ell)] \leq \mathrm{negl}\,.$$

- Compactness: There exists a polynomial $p(\lambda)$ such that the output of Eval has $p(\lambda)$ bits, and $p(\lambda)$ is independent of size of f and the number of inputs.

We provide the definition of language family with hard membership property.

Definition 3.4. A language family with hard membership is a tuple of three algorithms $(\mathsf{Gen}, \mathsf{Sam}, \mathsf{R})$ with the following syntax:

- Gen: On input 1^λ outputs some public parameter pp.
- Sam: On input pp and a bit $b \in \{0,1\}$, it uniformly samples a YES/NO instance of of the language depending on the bit b.
- R: On inputs (pp, x, w) outputs a bit which denotes whether x belongs to the language or not.

We require the following properties:

- Correctness: For any $\mathsf{pp} \leftarrow \mathsf{Gen}(1^\lambda)$ and any $x \leftarrow \mathsf{Sam}(1)$ (respectively $x \leftarrow \mathsf{Sam}(1)$), there exists (respectively, does not exist) a witness w such that $\mathsf{R}(\mathsf{pp}, x, w) = 1$ (respectively, $\mathsf{R}(\mathsf{pp}, x, w) = 0$).
- Security: If $x_0 \leftarrow \mathsf{Sam}(0)$ and $x_1 \leftarrow \mathsf{Sam}(1)$ then $x_0 \overset{c}{\approx} x_1$.
- Uniqueness: For any $\mathsf{pp} \leftarrow \mathsf{Gen}(1^\lambda)$, any $x \leftarrow \mathsf{Sam}(1)$, and any pair of witnesses w and w' if $\mathsf{R}(\mathsf{pp}, x, w) = \mathsf{R}(\mathsf{pp}, x, w') = 1$ then $w = w'$.

We recall the definition of indistinguishability obfuscation (iO) from [BGI+01].

Definition 3.5. A PPT algorithm Obf is an indistinguishability obfuscator for a circuit family $\mathcal{C}_\lambda$ with input space $\{0,1\}^{\ell(\lambda)}$ if:

- **Correctness:** For every circuit $C \in \mathcal{C}_\lambda$ and every input $x \in \{0,1\}^{\ell(\lambda)}$ we have:

$$\Pr[C(x) = C'(x) : C' \leftarrow \mathsf{Obf}(C)] = 1,$$

  where the probability is taken over the randomness of Obf algorithm.

- **Security:** For any PPT attacker $\mathcal{A}$ and for any two functionally equivalent circuits $C_0 \in \mathcal{C}_\lambda$ and $C_1 \in \mathcal{C}_\lambda$ such that $|C_0| = |C_1|$, it holds that:

$$|\Pr[\mathcal{A}(\lambda, C_0) = 1] - \Pr[\mathcal{A}(\lambda, C_1) = 1]| \leq \mathrm{negl}(\lambda).$$

We recall the definition of piO for a class of samplers from [CLTV15].

Definition 3.6. We say that piO is an indistinguishability obfuscator for a class of samplers $\mathcal{S}$ over the (randomized) circuit family $\mathcal{C}$ if is satisfies the following properties:

- On input a circuit C and security parameter λ, outputs a deterministic circuit $\overline{C}$ of size $\mathrm{poly}(|C|, \lambda)$.
- For every PPT attacker $\mathcal{A}$, every circuit C, and string str, consider the following experiments:
  - $\mathsf{Exp}^0_{\mathcal{A}}(C, \mathsf{str})$: $\mathcal{A}$ participates in an unbounded number of iterations, and in iteration i, $\mathcal{A}$ chooses an x_i; if x_i is equal to any of the previously chosen input x_j (for $j < i$) abort. Otherwise, $\mathcal{A}$ gets $C(x_i; r_i)$ using fresh randomness r_i. Finally, $\mathcal{A}$ outputs a bit b.

– $\mathsf{Exp}^1_{\mathcal{A}}(C, \mathsf{str})$: Let $\overline{C} = \mathsf{piO}(C; r)$. Run $\mathcal{A}$ as in the previous experiment except that in each iteration, provide $\mathcal{A}$ with $\overline{C}(x_i)$.

We require that for any PPT attacker $\mathcal{A}$, every circuit C, and and every polynomial-length string str it holds that $\left|\Pr[\mathcal{A}_{\mathsf{Exp}_0} = 1] - \Pr[\mathcal{A}_{\mathsf{Exp}_1} = 1]\right| \leq \mathrm{negl}$.

– For every PPT attacker $\mathcal{A}$ and every sampler $S \in \mathcal{S}$, if $(C_0, C_1, \mathsf{str}) \leftarrow S$ we have

$$\left| \Pr[\mathcal{A}(C_0, C_1, \mathsf{piO}(C_0), \mathsf{str}) = 1] - \Pr[\mathcal{A}(C_0, C_1, \mathsf{piO}(C_0), \mathsf{str}) = 1]\right| \leq \mathrm{negl}\,.$$

Symmetric MMap. Let R be a ring and let 0_R be the zero element for the ring.

Definition 3.7. (Symmetric Multilinear Map.) A symmetric multilinear map (MMap) consists of the following polynomial-time algorithms:

- $\mathsf{Setup}(1^\lambda, 1^N)$: Takes as input the security parameter λ and the degree of multilinearity N, and outputs a public parameter pp.
- $\mathsf{Encode}(\mathsf{pp}, a, i)$: Takes as input the public parameter pp, a plaintext element $a \in R$ and a level $i \in [0, N]$, and outputs the encoding $[a]_i$.
- $\mathsf{Add}(\mathsf{pp}, [a]_i, [b]_i)$: Takes as input the public parameter pp and two encodings $[a]_i$ and $[b]_i$ corresponding to a valid level $i \in [0, N]$, and outputs either the encoding $[a + b]_i$ or $\perp$ (in case one or more of the input encodings are invalid).
- $\mathsf{Inv}(\mathsf{pp}, [a]_i)$: Takes as input the public parameter pp and an encoding $[a]_i$ corresponding to a valid level $i \in [0, N]$, and outputs either the encoding $[(-1)_R \cdot a]_i$ or $\perp$ (in case the input encoding is invalid).
- $\mathsf{Mult}(\mathsf{pp}, [a]_{i_1}, [b]_{i_2})$: Takes as input the public parameter pp and two encodings $[a]_{i_1}$ and $[b]_{i_2}$ such that $i_1 + i_2 \leq N$, and outputs either the encoding $[a.b]_{i_1+i_2}$ or $\perp$ (in case one or more of the input encodings are invalid).
- $\mathsf{Ext}(\mathsf{pp}, [a]_i)$: Takes as input the public parameter pp and an encoding $[a]_i$ corresponding to a valid level i, and outputs either $s \in \{0,1\}^\lambda$ or $\perp$ (in case the input encoding is invalid).
- $\mathsf{ZeroTest}(\mathsf{pp}, [a]_i)$: Takes as input the public parameter pp and an encoding $[a]_i$ corresponding to a valid level i, and outputs $b \in \{0, 1, \perp\}$, where:
  - 1 indicates that $a = 0_R$.
  - 0 indicates that $a \neq 0_R$.
  - $\perp$ indicates that the encoding is invalid.

Remark 3.8. Note that equality-checking of encodings at any valid level follows implicitly from the addition, inversion and zero-test operations at the same level.

Asymmetric MMap. Let R be a ring and let 0_R be the zero element for the ring. Also let $\mathbf{0}$ and $\mathbf{n}$ be the all-zeroes and all-ones vectors of length N, respectively. A vector $\mathbf{i} \in \mathbb{Z}^n$ is said to represent "valid level" if $\mathbf{0} \leq \mathbf{i} \leq \mathbf{n}$, where comparison of vectors is performed component-wise.

Definition 3.9. (Asymmetric Multilinear Map.) An asymmetric multilinear map (MMap) consists of the following polynomial-time algorithms:

- $\mathsf{Setup}(1^\lambda, 1^N)$: Takes as input the security parameter λ and the degree of multilinearity N, and outputs a public parameter pp.
- $\mathsf{Encode}(\mathsf{pp}, a, \mathbf{i})$: Takes as input the public parameter pp, a plaintext element $a \in R$ and a valid level-vector $\mathbf{i}$, and outputs the encoding $[a]_{\mathbf{i}}$.

- $\mathsf{Add}(\mathsf{pp}, [a]_{\mathbf{i}}, [b]_{\mathbf{i}})$: Takes as input the public parameter pp and two encodings $[a]_{\mathbf{i}}$ and $[b]_{\mathbf{i}}$ corresponding to a valid level-vector $\mathbf{i}$, and outputs either the encoding $[a+b]_{\mathbf{i}}$ or $\perp$ (in case one or more of the input encodings are invalid).
- $\mathsf{Inv}(\mathsf{pp}, [a]_{\mathbf{i}})$: Takes as input the public parameter pp and an encoding $[a]_{\mathbf{i}}$ corresponding to a valid level-vector $\mathbf{i}$, and outputs either the encoding $[(-1)_R \cdot a]_{\mathbf{i}}$ or $\perp$ (in case the input encoding is invalid).
- $\mathsf{Mult}(\mathsf{pp}, [a]_{\mathbf{i}_1}, [b]_{\mathbf{i}_2})$: Takes as input the public parameter pp and two encodings $[a]_{\mathbf{i}_1}$ and $[b]_{\mathbf{i}_2}$ such that $\mathbf{i}_1 + \mathbf{i}_2 \leq \mathbf{n}$, and outputs either the encoding $[a.b]_{\mathbf{i}_1+\mathbf{i}_2}$ or $\perp$ (in case one or more of the input encodings are invalid).
- $\mathsf{Ext}(\mathsf{pp}, [a]_{\mathbf{i}})$: Takes as input the public parameter pp and an encoding $[a]_{\mathbf{i}}$ corresponding to a valid level-vector $\mathbf{i}$, and outputs either $s \in \{0,1\}^\lambda$ or $\perp$ (in case the input encoding is invalid).
- $\mathsf{ZeroTest}(\mathsf{pp}, [a]_{\mathbf{i}})$: Takes as input the public parameter pp and an encoding $[a]_{\mathbf{i}}$ corresponding to a valid level-vector $\mathbf{i}$, and outputs $b \in \{0, 1, \perp\}$, where:
  - 1 indicates that $a = 0_R$.
  - 0 indicates that $a \neq 0_R$.
  - $\perp$ indicates that the encoding is invalid.

Remark 3.10. Again, note that equality-checking of encodings at any valid level follows implicitly from the addition, inversion and zero-test operations at the same level.

Comparison with "Dream Version" Definitions. Note that our definition of symmetric/asymmetric multilinear maps almost identically achieves the full "dream version" of features that are defined and discussed in [GGH13a] and thus should be usable in any application of multilinear maps. The only technical difference is that we permit sampling an encoding for a specific plaintext value $a \in R$, while the authors of [GGH13a] provide an algorithm to sample a random $a \leftarrow R$ along with its encoding.

4 Constructing an SXDH-Hard Asymmetric MMap

In this section, we show a how to construct an SXDH-hard asymmetric MMap. Recall that the SXDH assumption over a degree-N asymmetric MMap requires that the following indistinguishability holds for any level-vector $\mathbf{i}$ such that $\mathbf{0} < \mathbf{i} \leq \mathbf{n}$:

$$\left([\alpha_0]_{\mathbf{i}}, [\alpha_1]_{\mathbf{i}}, [\alpha_0 \cdot \alpha_1]_{\mathbf{i}}\right) \quad \stackrel{c}{\approx} \quad \left([\alpha_0]_{\mathbf{i}}, [\alpha_1]_{\mathbf{i}}, [\alpha^*]_{\mathbf{i}}\right),$$

where $\alpha_0, \alpha_1, \alpha^* \leftarrow \mathbb{Z}_q$.

We show how to construct an SXDH-hard asymmetric MMap given the following cryptoprimitives:

- A probabilistic-iO scheme $\mathsf{piO} = (\mathsf{piO.Obf}, \mathsf{piOEval})$.
- A fully-homomorphic encryption scheme $\mathsf{FHE} = (\mathsf{FHE.Gen}, \mathsf{FHE.Enc}, \mathsf{FHE.Dec}, \mathsf{FHE.Eval})$ such that the message space is $\mathbb{Z}_q$ for some prime $q = \mathrm{poly}(\lambda)$ (λ being the security parameter).
- A simulation-extractable NIZK argument system $\mathsf{NIZK} = (\mathsf{NIZK.Setup}, \mathsf{NIZK.Prove}, \mathsf{NIZK.Verify})$.
- A pair of sets $(\mathcal{X}, \mathcal{L})$ such that $\mathcal{L} \subset \mathcal{X}$ and:
  1. Given $x \in \mathcal{X}$ it is *computationally hard* to decide if $x \in \mathcal{L}$.
  2. For each $y \in \mathcal{L}$, there exists a *unique* witness wit_y for the statement $y \in \mathcal{L}$.

  We define a relation $\mathsf{R}_{\mathcal{L}}$ as follows: $(y, \mathsf{wit}_y) \in R_{\mathcal{L}}$ if and only if wit_y is a witness for the statement $y \in \mathcal{L}$.
- A pairing-free DDH-hard group $\mathbb{G}$ of prime order q, where $\mathcal{D}_{\mathrm{DDH},\mathbb{G}}$ denotes the distribution of all valid DDH samples over the group $\mathbb{G}$.

4.1 Encodings

Level-Zero Encodings. In our construction, level-$\mathbf{0}$ encodings are treated slightly different from encodings at other "non-zero" levels. In particular, our level-$\mathbf{0}$ encodings are equipped with their own set of algorithms for encoding, manipulation, extraction and zero-testing. We informally mention these for the sake of completeness.

The level-$\mathbf{0}$ encoding of a plaintext element $a \in \mathbb{Z}_q$ is a itself. Adding/multiplying two level-$\mathbf{0}$ encodings (equivalently, additively inverting a level-$\mathbf{0}$ encoding) is simply done via addition/multiplication (equivalently, additive inversion) in $\mathbb{Z}_q$. Multiplying a level-$\mathbf{0}$ encoding with any other encoding at some level $\mathbf{i}$ should result in an encoding at level-$\mathbf{i}$. This is implemented with a shift-and-add algorithm built on top of the standard encoding addition algorithm described subsequently in Section 4.2. Finally, extracting a level-$\mathbf{0}$ encoding $a \in \mathbb{Z}_q$ outputs g^a, where g is a fixed generator for the group $\mathbb{G}$. As will be clear later, this is consistent with the extraction algorithm for any other level $\mathbf{i}$. Zero-testing follows trivially.

We omit formal descriptions of operations for level-$\mathbf{0}$ encodings to ease notation and focus on the more interesting cases at "non-zero" encoding levels.

Level-i Encodings. We now describe the procedure of encoding a plaintext element at any level $\mathbf{i}$ such that $\mathbf{0} < \mathbf{i} \leq \mathbf{n}$. An encoding of a plaintext element $a \in \mathbb{Z}_q$ at level-vector $\mathbf{i}$ is a tuple of the form: $(\mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, \pi)$, where:

- ct_0 and ct_1 are FHE encryptions of the tuple $(a, 0, 0, 0, \mathbf{i})$ under the public key-secret key pairs $(\mathsf{pk}_0, \mathsf{sk}_0)$ and $(\mathsf{pk}_1, \mathsf{sk}_1)$ respectively.
- π is a NIZK proof for the statement $\mathsf{st} = (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, y \in \mathcal{X})$ under the NP language L described in Figure 1, where $y \in \mathcal{X}$ is available as part of the public parameter for our MMap scheme.

Note that the additional zero entries encrypted in each encoding appear redundant, but will be useful later in the proof of security.

Language L:

Statement: The statement st is as follows: $\mathsf{st} = (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, y \in \mathcal{X})$

Witness: The witness has one of the following three forms:

1. $\mathsf{wit} = (\mathsf{sk}_0, \mathsf{sk}_1)$
2. $\mathsf{wit} = (\mathsf{m}, \mathsf{r}_0, \mathsf{r}_1)$
3. $\mathsf{wit} = \mathsf{wit}_y$

Relation: $\mathsf{R}(\mathsf{st}, \mathsf{wit}) = 1$ if and only if:

- **Either** $\mathsf{wit} = (\mathsf{sk}_0, \mathsf{sk}_1)$ and we have:
  1. $(\mathsf{pk}_0, \mathsf{sk}_0), (\mathsf{pk}_1, \mathsf{sk}_1) \in \mathcal{K}_{\mathrm{FHE}}$.
  2. $\mathsf{FHE.Dec}(\mathsf{sk}_0, \mathsf{ct}_0) = \mathsf{FHE.Dec}(\mathsf{sk}_1, \mathsf{ct}_1) = (\mathsf{m}, 0, 0, 0, \mathbf{i})$ for some $\mathsf{m} \in \mathbb{Z}_q$.
- **Or** $\mathsf{wit} = (\mathsf{m}, \mathsf{r}_0, \mathsf{r}_1)$ and we have:
  1. $\mathsf{m} \in \mathbb{Z}_q$.
  2. For each $b \in \{0, 1\}$, $\mathsf{FHE.Enc}(\mathsf{pk}_b, (\mathsf{m}, 0, 0, 0, \mathbf{i}); \mathsf{r}_b) = \mathsf{ct}_b$.
- **Or** $\mathsf{wit} = \mathsf{wit}_y$ and we have: $(y, \mathsf{wit}_y) \in \mathsf{R}_{\mathcal{L}}$.

Figure 1: Let $\mathcal{K}_{\mathrm{FHE}}$ be the set of all valid key pairs under the FHE scheme. We define an NP language L as in the figure above. For simplicity, L parameterized by N – the number of levels for our MMap scheme.

4.2 Addition of Encodings

We now describe the procedure for adding two encodings. Suppose we have two encodings at the same level $\mathbf{i}$ of the form:

$$(\mathsf{ct}_{1,0}, \mathsf{ct}_{1,1}, \mathbf{i}, \pi_1), (\mathsf{ct}_{2,0}, \mathsf{ct}_{2,1}, \mathbf{i}, \pi_2).$$

Conceptually, we add these two encodings by exploiting the fully-homomorphic nature of the encryption scheme. More concretely, we homomorphically evaluate the circuit $\mathsf{C}_{\mathsf{Add,FHE}}$ (described in Figure 2) on the corresponding ciphertext components of the two input encodings to generate the ciphertext components for the output encoding. We also generate a proof for the validity of the resulting encoding using the tuple of secret keys $(\mathsf{sk}_0, \mathsf{sk}_1)$ as witness.

Inputs: A tuple of the form $((\{a_{1,\ell}\}_{\ell\in[0,3]}, \mathbf{i}_1), ((\{a_{2,\ell}\}_{\ell\in[0,3]}, \mathbf{i}_2))$, where for each $\ell \in [0,3]$, we have $a_{1,\ell}, a_{2,\ell} \in \mathbb{Z}_q$.

Computation: Proceed as follows:

1. If $\mathbf{i}_1 \neq \mathbf{i}_2$ or $\mathbf{i}_1 \leq \mathbf{0}$ or $\mathbf{i}_1 > \mathbf{n}$, output $\perp$.
2. Else, output $(\{a_{1,\ell} + a_{2,\ell}\}_{\ell\in[0,3]}, \mathbf{i}_1)$.

Figure 2: Circuit $\mathsf{C}_{\mathsf{Add,FHE}}$

For technical reasons that are relevant to the proof of security, we check: (a) the validity of the proofs π_1 and π_2 for the language L that are provided as part of the input encodings, and (b) the "consistency" of the FHE ciphertexts that are provided as part of the input encodings, i.e., whether the ciphertexts encrypt the same underlying plaintext, and whether the plaintext is formatted as per the specifications of the scheme. While "validity" is publicly verifiable, verifying "consistency" requires knowledge of the secret keys sk_0 and sk_1.

Figure 3 details the operation of the encoding-addition circuit $\mathsf{C}_{\mathsf{Add,MMap}}$. Of course, it embeds multiple secrets, including the secret keys sk_0 and sk_1, as well as the extraction trapdoor $\mathsf{t}_{\mathsf{ext}}$ for the NIZK. Hence, the circuit is not made public as is; we only make available an obfuscated version of the circuit obtained by running the evaluation algorithm of the probabilistic iO scheme piO on it.

We emphasize that when the element $y \in \mathcal{X}$ is a non-member for the language $\mathcal{L}$, then under a binding crs, the "**If**" condition in step 4 of $\mathsf{C}_{\mathsf{Add,MMap}}$ is never satisfied. This follows from the perfect extractability guarantee of the NIZK proof system. However, the condition may be satisfied during some hybrid in the proof of security, under a hiding crs when the element $y \in \mathcal{X}$ is a member for the language $\mathcal{L}$.

4.3 Inversion of Encodings

We now describe the procedure for additively inverting an encoding. Suppose we have an encoding at the level $\mathbf{i}$ of the form: $(\mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, \pi)$. Again, conceptually, we additively invert this encoding by exploiting the fully-homomorphic nature of the encryption scheme. More concretely, we homomorphically evaluate the circuit $\mathsf{C}_{\mathsf{Inv,FHE}}$ (described in Figure 4) on the corresponding ciphertext components of the input encoding to generate the ciphertext components for the output encoding. We also generate a proof for the validity of the resulting encoding using the tuple of secret keys $(\mathsf{sk}_0, \mathsf{sk}_1)$ as witness.

Similar to the addition procedure, for technical reasons that are relevant to the proof of security, we check the validity and consistency of the input encodings. Figure 5 details the operation of the encoding-inversion circuit $\mathsf{C}_{\mathsf{Inv,MMap}}$. Since it embeds the same set of secrets as the addition circuit, the circuit $\mathsf{C}_{\mathsf{Inv,MMap}}$ is not made public as is; we only make available an obfuscated version of the circuit obtained by running the evaluation algorithm of the probabilistic iO scheme piO on it.

Finally, we emphasize that when the element $y \in \mathcal{X}$ is a non-member for the language $\mathcal{L}$, then under a binding crs, the "**If**" condition in step 4 of $\mathsf{C}_{\mathsf{Add,MMap}}$ is never satisfied. This follows from the perfect extractability guarantee of the NIZK proof system. However, the condition may be satisfied during some hybrid in the proof of security, under a hiding crs when the element $y \in \mathcal{X}$ is a member for the language $\mathcal{L}$.

Encoding Addition Circuit: $\mathsf{C}_{\mathsf{Add,MMap}}$

Inputs: A tuple of the form: $((\mathsf{ct}_{1,0}, \mathsf{ct}_{1,1}, \mathbf{i}, \pi_1), (\mathsf{ct}_{2,0}, \mathsf{ct}_{2,1}, \mathbf{i}, \pi_2), y \in \mathcal{X})$, where the proofs π_1 and π_2 are over the language L.

Outputs: Either a tuple of the form $(\mathsf{ct}_0^*, \mathsf{ct}_1^*, \mathbf{i}, \pi^*)$, or $\perp$.

Computation: Proceed as follows:

1. **If** any of the following is true:
   (a) $\mathsf{NIZK.Verify}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_{1,0}, \mathsf{ct}_{1,1}, \mathbf{i}, y \in \mathcal{X}), \pi_1) = 0$.
   (b) $\mathsf{NIZK.Verify}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_{2,0}, \mathsf{ct}_{2,1}, \mathbf{i}, y \in \mathcal{X}), \pi_2) = 0$.

   then output $\perp$.
2. Recover the following:
$$(\{a_{1,0,\ell}\}_{\ell\in[0,3]}, \mathbf{i}_{1,0}) = \mathsf{FHE.Dec}(\mathsf{sk}_0, \mathsf{ct}_{1,0}) \quad , \quad (\{a_{1,1,\ell}\}_{\ell\in[0,3]}, \mathbf{i}_{1,1}) = \mathsf{FHE.Dec}(\mathsf{sk}_1, \mathsf{ct}_{1,1}),$$
$$(\{a_{2,0,\ell}\}_{\ell\in[0,3]}, \mathbf{i}_{2,0}) = \mathsf{FHE.Dec}(\mathsf{sk}_0, \mathsf{ct}_{2,0}) \quad , \quad (\{a_{2,1,\ell}\}_{\ell\in[0,3]}, \mathbf{i}_{2,1}) = \mathsf{FHE.Dec}(\mathsf{sk}_1, \mathsf{ct}_{2,1}).$$
3. Compute the following:
$$\mathsf{ct}_0^* = \mathsf{FHE.Eval}(\mathsf{pk}_0, \mathsf{ct}_{1,0}, \mathsf{ct}_{2,0}, \mathsf{C}_{\mathsf{Add,FHE}}) \quad , \quad \mathsf{ct}_1^* = \mathsf{FHE.Eval}(\mathsf{pk}_1, \mathsf{ct}_{1,1}, \mathsf{ct}_{2,1}, \mathsf{C}_{\mathsf{Add,FHE}}).$$
4. **If** any of the following is true:
   (a) $a_{1,0,0} \neq a_{1,1,0}$ or $a_{2,0,0} \neq a_{2,1,0}$.
   (b) There exists some $(t, b, \ell) \in \{1, 2\} \times \{0, 1\} \times \{1, 2, 3\}$ such that $a_{t,b,\ell} \neq 0$.
   (c) There exists some $(t, b) \in \{1, 2\} \times \{0, 1\}$ such that $\mathbf{i}_{t,b} \neq \mathbf{i}$.

   then proceed as follows:
   (a) Extract $\mathsf{wit}_y = \mathsf{NIZK.Ext}(\mathsf{t}_{\mathsf{ext}}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_{1,0}, \mathsf{ct}_{1,1}, \mathbf{i}, y), \pi_1)$.
   (b) **If** $\mathsf{R}_{\mathcal{L}}(\mathsf{wit}_y, y) = 0$, output $\perp$.
   (c) **Else**, generate $\pi^* \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0^*, \mathsf{ct}_1^*, \mathbf{i}, y), \mathsf{wit}_y)$.
5. **Else**, generate $\pi^* \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0^*, \mathsf{ct}_1^*, \mathbf{i}, y), (\mathsf{sk}_0, \mathsf{sk}_1))$.
6. Output $(\mathsf{ct}_0^*, \mathsf{ct}_1^*, \mathbf{i}, \pi^*)$.

Figure 3: Let $\mathsf{crs} \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$. We now assume that crs is in binding mode. Let $\mathsf{NIZK.Ext}$ be the extraction algorithm for NIZK. Let $\mathsf{t}_{\mathsf{ext}}$ be the extraction trapdoor corresponding to crs. Additionally, let $(\mathsf{pk}_0, \mathsf{sk}_0), (\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathsf{FHE.Gen}(1^\lambda)$. We define a probabilistic circuit $\mathsf{C}_{\mathsf{Add,MMap}}$ (which has $\mathsf{t}_{\mathsf{ext}}$ and the key pairs $(\mathsf{pk}_0, \mathsf{sk}_0)$ and $(\mathsf{pk}_1, \mathsf{sk}_1)$ hard-wired into it) as above.

Inputs: A tuple of the form $((a_0, a_1, a_2, a_3, \mathbf{i}))$ where for each $\ell \in [0, 3]$, we have $a_\ell \in \mathbb{Z}_q$.

Computation: If $\mathbf{i} \leq \mathbf{0}$ or $\mathbf{i} > N$, output $\perp$ else output $(-a_0, -a_1, -a_2, -a_3, \mathbf{i})$.

Figure 4: Circuit $\mathsf{C}_{\mathsf{Inv,FHE}}$

4.4 Multiplication of Encodings

We now describe the procedure for multiplying two encodings at levels $\mathbf{i}_1$ and $\mathbf{i}_2$, respectively such that $\mathbf{i}_1 + \mathbf{i}_2 \leq \mathbf{n}$. Suppose we have two encodings of the form:

$$(\mathsf{ct}_{1,0}, \mathsf{ct}_{1,1}, \mathbf{i}_1, \pi_1), (\mathsf{ct}_{2,0}, \mathsf{ct}_{2,1}, \mathbf{i}_2, \pi_2).$$

Conceptually, we multiply these two encodings by again exploiting the fully-homomorphic nature of the encryption scheme. More concretely, we homomorphically evaluate the circuit $\mathsf{C}_{\mathsf{Mult,FHE}}$ (described in Figure 6) on the corresponding ciphertext components of the two input encodings to generate the ciphertext components for the output encoding. We also generate a proof for the validity of the resulting encoding using the tuple of secret keys $(\mathsf{sk}_0, \mathsf{sk}_1)$ as witness.

Similar to the addition and inversion procedures, for technical reasons that are relevant to the proof of security, we check the validity and consistency of the input encodings. Figure 7 details the operation of the encoding-multiplication

Encoding Inversion Circuit: $\mathsf{C}_{\mathsf{Inv,MMap}}$

Inputs: A tuple of the form: $(\{\mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, \pi, y \in \mathcal{X})$, where the proof π is over the language L.

Outputs: Either a tuple of the form $(\mathsf{ct}_0^*, \mathsf{ct}_1^*, \mathbf{i}, \pi^*)$, or $\perp$.

Computation: Proceed as follows:

1. If $\mathsf{NIZK.Verify}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, y), \pi) = 0$, then output $\perp$.
2. Recover the following:
$$(\{a_{0,\ell}\}_{\ell\in[0,3]}, \mathbf{i}_0) = \mathsf{FHE.Dec}(\mathsf{sk}_0, \mathsf{ct}_0) \quad , \quad (\{a_{1,\ell}\}_{\ell\in[0,3]}, \mathbf{i}_1) = \mathsf{FHE.Dec}(\mathsf{sk}_1, \mathsf{ct}_1).$$
3. Compute the following:
$$\mathsf{ct}_0^* = \mathsf{FHE.Eval}(\mathsf{pk}_0, \mathsf{ct}_0, \mathsf{C}_{\mathsf{Inv,FHE}}) \quad , \quad \mathsf{ct}_1^* = \mathsf{FHE.Eval}(\mathsf{pk}_1, \mathsf{ct}_1, \mathsf{C}_{\mathsf{Inv,FHE}}).$$
4. If any of the following is true:
   (a) $a_{0,0} \neq a_{0,1}$.
   (b) There exists some $(b, \ell) \in \{0, 1\} \times \{1, 2, 3\}$ such that $a_{b,\ell} \neq 0$.
   (c) There exists some $b \in \{0, 1\}$ such that $\mathbf{i}_b \neq \mathbf{i}$.

   then proceed as follows:
   (a) Extract $\mathsf{wit}_y = \mathsf{NIZK.Ext}(\mathsf{t}_{\mathsf{ext}}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, y), \pi)$.
   (b) **If** $\mathsf{R}_{\mathcal{L}}(\mathsf{wit}_y, y) = 0$, output $\perp$.
   (c) **Else**, generate $\pi^* \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0^*, \mathsf{ct}_1^*, \mathbf{i}, y), \mathsf{wit}_y)$.
5. **Else**, generate $\pi^* \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0^*, \mathsf{ct}_1^*, \mathbf{i}, y), (\mathsf{sk}_0, \mathsf{sk}_1))$.
6. Output $(\mathsf{ct}_0^*, \mathsf{ct}_1^*, \mathbf{i}, \pi^*)$.

Figure 5: Let $\mathsf{crs} \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$. We now assume that crs is in binding mode. Let $\mathsf{NIZK.Ext}$ be the extraction algorithm for NIZK. Let $\mathsf{t}_{\mathsf{ext}}$ be the extraction trapdoor corresponding to crs. Additionally, let $(\mathsf{pk}_0, \mathsf{sk}_0), (\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathsf{FHE.Gen}(1^\lambda)$. We define a probabilistic circuit $\mathsf{C}_{\mathsf{Inv,MMap}}$ (which has $\mathsf{t}_{\mathsf{ext}}$ and the key pairs $(\mathsf{pk}_0, \mathsf{sk}_0)$ and $(\mathsf{pk}_1, \mathsf{sk}_1)$ hard-wired into it) as above.

Inputs: A tuple of the form $((\{a_{1,\ell}\}_{\ell\in[0,3]}, \mathbf{i}_1), ((\{a_{2,\ell}\}_{\ell\in[0,3]}, \mathbf{i}_2))$, where for each $\ell \in [0, 3]$, we have $a_{1,\ell}, a_{2,\ell} \in \mathbb{Z}_q$.

Computation: Proceed as follows:

- If $\mathbf{i}_1 \leq \mathbf{0}$ or $\mathbf{i}_2 \leq \mathbf{0}$ or $\mathbf{i}_1 + \mathbf{i}_2 > \mathbf{n}$, output $\perp$.
- Else if $(a_{1,1}, a_{1,2}, a_{1,3}) = (0, 0, 0)$ then output $((\{a_{1,0} \cdot a_{2,\ell}\}_{\ell\in[0,3]}, i_1, j_2))$.
- Else if $(a_{2,1}, a_{2,2}, a_{2,3}) = (0, 0, 0)$ then output $((\{a_{2,0} \cdot a_{1,\ell}\}_{\ell\in[0,3]}, i_1, j_2))$.
- Else output $\perp$.

Figure 6: Circuit $\mathsf{C}_{\mathsf{Mult,FHE}}$

circuit $\mathsf{C}_{\mathsf{Mult,MMap}}$. Once again, it embeds multiple secrets, including the secret keys sk_0 and sk_1, as well as the extraction trapdoor $\mathsf{t}_{\mathsf{ext}}$ for the NIZK. Hence, the circuit is not made public as is; we only make available an obfuscated version of the circuit obtained by running the evaluation algorithm of the probabilistic iO scheme piO on it.

We emphasize that when the element $y \in \mathcal{X}$ is a non-member for the language $\mathcal{L}$, then under a binding crs, the "**If**" condition in step 4 of $\mathsf{C}_{\mathsf{Mult,MMap}}$ is never satisfied. This follows from the perfect extractability guarantee of the NIZK proof system. However, the condition may be satisfied during some hybrid in the proof of security, under a hiding crs when the element $y \in \mathcal{X}$ is a member for the language $\mathcal{L}$.

This in turn guarantees that when the element $y \in \mathcal{X}$ is a non-member for the language $\mathcal{L}$, then under a binding crs, no encoding (even adversarially generated) can have "auxiliary" FHE ciphertexts encrypting other than the plaintext element 0. Hence, during multiplication, we do not need to worry about computing and adding "cross-product" terms across the input ciphertexts.

Encoding Multiplication Circuit: $\mathsf{C}_{\mathsf{Mult},\mathbf{MMap}}$

Inputs: A tuple of the form: $((\mathsf{ct}_{1,0}, \mathsf{ct}_{1,1}, \mathbf{i}_1, \pi_1), (\mathsf{ct}_{2,0}, \mathsf{ct}_{2,1}, \mathbf{i}_2, \pi_2), y \in \mathcal{X})$, where the proofs π_1 and π_2 are over the language L.

Outputs: Either a tuple of the form $(\mathsf{ct}_0^*, \mathsf{ct}_1^*, \mathbf{i}_1 + \mathbf{i}_2, \pi^*)$, or $\perp$.

Computation: Proceed as follows:

1. **If** any of the following is true:
   (a) $\mathsf{NIZK.Verify}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_{1,0}, \mathsf{ct}_{1,1}, \mathbf{i}_1, y \in \mathcal{X}), \pi_1) = 0$.
   (b) $\mathsf{NIZK.Verify}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_{2,0}, \mathsf{ct}_{2,1}, \mathbf{i}_2, y \in \mathcal{X}), \pi_2) = 0$.

   then output $\perp$.
2. Recover the following:
$$(\{a_{1,0,\ell}\}_{\ell\in[0,3]}, \mathbf{i}_{1,0}) = \mathsf{FHE.Dec}(\mathsf{sk}_0, \mathsf{ct}_{1,0}) \quad , \quad (\{a_{1,1,\ell}\}_{\ell\in[0,3]}, \mathbf{i}_{1,1}) = \mathsf{FHE.Dec}(\mathsf{sk}_1, \mathsf{ct}_{1,1}),$$
$$(\{a_{2,0,\ell}\}_{\ell\in[0,3]}, \mathbf{i}_{2,0}) = \mathsf{FHE.Dec}(\mathsf{sk}_0, \mathsf{ct}_{2,0}) \quad , \quad (\{a_{2,1,\ell}\}_{\ell\in[0,3]}, \mathbf{i}_{2,1}) = \mathsf{FHE.Dec}(\mathsf{sk}_1, \mathsf{ct}_{2,1}).$$
3. Compute the following:
$$\mathsf{ct}_0^* = \mathsf{FHE.Eval}(\mathsf{pk}_0, \mathsf{ct}_{1,0}, \mathsf{ct}_{2,0}, \mathsf{C}_{\mathsf{Mult,FHE}}) \quad , \quad \mathsf{ct}_1^* = \mathsf{FHE.Eval}(\mathsf{pk}_1, \mathsf{ct}_{1,1}, \mathsf{ct}_{2,1}, \mathsf{C}_{\mathsf{Mult,FHE}}).$$
4. **If** any of the following is true:
   (a) $a_{1,0,0} \neq a_{1,1,0}$ or $a_{2,0,0} \neq a_{2,1,0}$.
   (b) There exists some $(t, b, \ell) \in \{1, 2\} \times \{0, 1\} \times \{1, 2, 3\}$ such that $a_{t,b,\ell} \neq 0$.
   (c) There exists some $b \in \{0, 1\}$ such that $\mathbf{i}_{1,b} \neq \mathbf{i}_1$ or $\mathbf{i}_{2,b} \neq \mathbf{i}_2$.

   then proceed as follows:
   (a) Extract $\mathsf{wit}_y = \mathsf{NIZK.Ext}(\mathsf{t}_{\mathsf{ext}}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_{1,0}, \mathsf{ct}_{1,1}, \mathbf{i}, y), \pi_1)$.
   (b) **If** $\mathsf{R}_{\mathcal{L}}(\mathsf{wit}_y, y) = 0$, output $\perp$.
   (c) **Else**, generate $\pi^* \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0^*, \mathsf{ct}_1^*, \mathbf{i}_1 + \mathbf{i}_2, y), \mathsf{wit}_y)$.
5. **Else**, generate $\pi^* \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0^*, \mathsf{ct}_1^*, \mathbf{i}_1 + \mathbf{i}_2, y), (\mathsf{sk}_0, \mathsf{sk}_1))$.
6. Output $(\mathsf{ct}_0^*, \mathsf{ct}_1^*, \mathbf{i}_1 + \mathbf{i}_2, \pi^*)$.

Figure 7: Let $\mathsf{crs} \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$. We now assume that crs is in binding mode. Let $\mathsf{NIZK.Ext}$ be the extraction algorithm for NIZK. Let $\mathsf{t}_{\mathsf{ext}}$ be the extraction trapdoor corresponding to crs. Additionally, let $(\mathsf{pk}_0, \mathsf{sk}_0), (\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathsf{FHE.Gen}(1^\lambda)$. We define a probabilistic circuit $\mathsf{C}_{\mathsf{Mult,MMap}}$ (which has $\mathsf{t}_{\mathsf{ext}}$ and the key pairs $(\mathsf{pk}_0, \mathsf{sk}_0)$ and $(\mathsf{pk}_1, \mathsf{sk}_1)$ hard-wired into it) as in above.

4.5 Extraction and Zero-Testing

Extraction. We now describe the procedure for extracting a canonical string from an encoding at any given level-vector. Suppose we have an encodings at the level $\mathbf{i}$ of the form: $(\{\mathsf{ct}_{\ell,b}\}_{\ell\in\{0,1,2,3\},b\in\{0,1\}}, \mathbf{i}, \pi)$. The straightforward way of doing this is to decrypt the corresponding ciphertexts using sk_0 and sk_1, and outputting $g^* = g^{a_{0,0}} = g^{a_{1,0}}$, where g is a fixed group element embedded inside the extraction circuit. However, for technical reasons relevant to the proof of security, we sightly depart from this straightforward process.

More concretely, our extraction circuit embeds a four-tuple of group elements (g, h, g', h'), uses sk_0 to recover $a_{0,\ell}$ for each $\ell \in \{0, 1, 2, 3\}$, and computes

$$g^* = g^{a_{0,0}} \cdot h^{a_{0,1}} \cdot (g')^{a_{0,2}} \cdot (h')^{a_{0,3}}.$$

It is easy to see that when the encodings are honestly generated, i.e., when the "auxiliary" terms in the plaintext are all 0, this is functionally equivalent to the simpler extraction strategy described above. However, if one or more of the "auxiliary" plaintext terms is non-zero, the two processes are no longer equivalent. As it turns out, we exploit this fact in the proof of security.

Again, for technical reasons that are relevant to the proof of security, we also use the secret keys sk_0 and sk_1, as well as the NIZK extraction trapdoor $\mathsf{t}_{\mathsf{ext}}$, to check: (a) the validity of the proofs π_1 and π_2 for the language L that are provided as part of the input encodings, and (b) the "consistency" of the FHE ciphertexts that are provided as part of the

Extraction Circuit: $\mathsf{C}_{\mathsf{Ext,MMap}}$

Inputs: A tuple of the form: $((\mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, \pi), y \in \mathcal{X})$, where the proof π is over the language L.

Outputs: Either a group element g^* or $\perp$.

Computation: Proceed as follows:

1. If $\mathsf{NIZK.Verify}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, y), \pi) = 0$, then output $\perp$.
2. Recover the following:
$$(\{a_{0,\ell}\}_{\ell \in [0,3]}, \mathbf{i}_0) = \mathsf{FHE.Dec}(\mathsf{sk}_0, \mathsf{ct}_0) \quad , \quad (\{a_{1,\ell}\}_{\ell \in [0,3]}, \mathbf{i}_1) = \mathsf{FHE.Dec}(\mathsf{sk}_1, \mathsf{ct}_1).$$
3. Compute $g^* = g^{a_{0,0}}.h^{a_{0,1}}.(g')^{a_{0,2}}.(h')^{a_{0,3}}$.
4. If any of the following is true:
   (a) $a_{0,0} \neq a_{0,1}$.
   (b) There exists some $(b, \ell) \in \{0,1\} \times \{1,2,3\}$ such that $a_{b,\ell} \neq 0$.
   (c) There exists some $b \in \{0,1\}$ such that $\mathbf{i}_b \neq \mathbf{i}$.

   then proceed as follows:
   (a) Extract $\mathsf{wit}_y = \mathsf{NIZK.Ext}(\mathsf{t}_{\mathsf{ext}}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, y), \pi)$.
   (b) **If** $\mathsf{R}_{\mathcal{L}}(\mathsf{wit}_y, y) = 0$, output $\perp$.
5. **Else**, output g^*.

Figure 8: Let $\mathsf{crs} \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$. We now assume that crs is in binding mode. Let $\mathsf{NIZK.Ext}$ be the extraction algorithm for NIZK. Let $\mathsf{t}_{\mathsf{ext}}$ be the extraction trapdoor corresponding to crs. Additionally, let $(\mathsf{pk}_0, \mathsf{sk}_0), (\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathsf{FHE.Gen}(1^\lambda)$, and let $g, h, g', h' \in \mathbb{G}$ be arbitrary group elements. We define a deterministic circuit $\mathsf{C}_{\mathsf{Ext,MMap}}$ (which has $\mathsf{t}_{\mathsf{ext}}$, the key pairs $(\mathsf{pk}_0, \mathsf{sk}_0)$ and $(\mathsf{pk}_1, \mathsf{sk}_1)$ and the collection of group elements (g, h, g', h') hard-wired into it) as above.

input encodings, i.e., whether the "auxiliary" set of plaintext elements are all zero and whether both ciphertexts encrypt the same level.

Figure 8 details the operation of the extraction circuit $\mathsf{C}_{\mathsf{Ext,MMap}}$. As before, the circuit $\mathsf{C}_{\mathsf{Ext,MMap}}$ is not made public as is; we only make available an obfuscated version of the circuit obtained by running the evaluation algorithm of the probabilistic iO scheme piO on it.

Finally, we emphasize that when the element $y \in \mathcal{X}$ is a non-member for the language $\mathcal{L}$, then under a binding crs, the "**If**" condition in step 4 of $\mathsf{C}_{\mathsf{Ext,MMap}}$ is never satisfied. This follows from the perfect extractability guarantee of the NIZK proof system. However, the condition may be satisfied during some hybrid in the proof of security, under a hiding crs when the element $y \in \mathcal{X}$ is a member for the language $\mathcal{L}$.

Zero-Testing Encodings. Given the aforementioned extraction procedure, zero-testing an encoding at any given level is trivial. We simply apply the extraction procedure to the encoding, and check if the extracted group element g^* is equal to g^0 for any $g \in \mathbb{G}$.

4.6 The Overall Construction

We are now ready to formally summarize our construction of an SXDH-hard MMap.

- $\mathsf{Setup}(1^\lambda, N)$: On input the security parameter λ and the degree of multilinearity N, do the following:
  1. Sample (in binding mode) $(\mathsf{crs}, \mathsf{t}_{\mathsf{ext}}) \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$.
  2. Sample $(\mathsf{pk}_0, \mathsf{sk}_0), (\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathsf{FHE.Gen}(1^\lambda)$.
  3. Sample a non-member instance $y \leftarrow \mathcal{X} \setminus \mathcal{L}$.
  4. Compute $\mathsf{C}_{\mathsf{Add,piO}} = \mathsf{piO.Obf}(\mathsf{C}_{\mathsf{Add,MMap}})$, where $\mathsf{C}_{\mathsf{Add,MMap}}$ is as defined in Figure 3.
  5. Compute $\mathsf{C}_{\mathsf{Inv,piO}} = \mathsf{piO.Obf}(\mathsf{C}_{\mathsf{Inv,MMap}})$, where $\mathsf{C}_{\mathsf{Inv,MMap}}$ is as defined in Figure 5.
  6. Compute $\mathsf{C}_{\mathsf{Mult,piO}} = \mathsf{piO.Obf}(\mathsf{C}_{\mathsf{Mult,MMap}})$, where $\mathsf{C}_{\mathsf{Mult,MMap}}$ is as defined in Figure 7.

7. Compute $\mathsf{C}_{\mathsf{Ext,piO}} = \mathsf{piO.Obf}(\mathsf{C}_{\mathsf{Ext,MMap}})$, where $\mathsf{C}_{\mathsf{Ext,MMap}}$ is as defined in Figure 8.
8. Output the public parameter pp, where

$$\mathsf{pp} = \left(\mathsf{crs}, \mathsf{pk}_0, \mathsf{pk}_1, \mathsf{C}_{\mathsf{Add,piO}}, \mathsf{C}_{\mathsf{Inv,piO}}, \mathsf{C}_{\mathsf{Mult,piO}}, \mathsf{C}_{\mathsf{Ext,piO}}, y\right).$$

- $\mathsf{Encode}(\mathsf{pp}, a, [\mathbf{i}])$: On input the public parameter pp, a plaintext element $a \in \mathbb{Z}_q$ and $i \in [N]$ such that $i \leq j \leq N$, proceed as follows:

  1. Generate the following ciphertexts:

  $$\mathsf{ct}_0 = \mathsf{FHE.Enc}(\mathsf{pk}_0, (a, 0, 0, 0, \mathbf{i}); \mathsf{r}_0) \quad , \quad \mathsf{ct}_1 \leftarrow \mathsf{FHE.Enc}(\mathsf{pk}_1, (a, 0, 0, 0, \mathbf{i}); \mathsf{r}_1),$$

  where r_0 and r_1 denote uniformly sampled random coins.
  2. Generate the following proof:

  $$\pi \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, \mathsf{st}, \mathsf{wit}),$$

  where the statement st and the witness wit correspond to the NP language L (as defined in Figure 1) and are given as:

  $$\mathsf{st} = (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, y) \quad , \quad \mathsf{wit} = (a, \mathsf{r}_0, \mathsf{r}_1).$$

  3. Finally, output the encoding

  $$[a]_{[\mathbf{i}]} := (\mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, \pi).$$

- $\mathsf{Add}(\mathsf{pp}, [a_1]_{[\mathbf{i}]}, [a_2]_{[\mathbf{i}]})$: On input the public parameter pp and a pair of encodings

$$[a_1]_{[\mathbf{i}]} = (\mathsf{ct}_{1,0}, \mathsf{ct}_{1,1}, \mathbf{i}, \pi_1) \quad , \quad [a_2]_{[\mathbf{i}]} = (\mathsf{ct}_{2,0}, \mathsf{ct}_{2,1}, \mathbf{i}, \pi_2),$$

  output the encoding $[a_1 + a_2]_{[\mathbf{i}]} = (\mathsf{ct}^*_0, \mathsf{ct}^*_1, \mathbf{i}, \pi^*)$, where

$$(\mathsf{ct}^*_0, \mathsf{ct}^*_1, \mathbf{i}, \pi^*) = \mathsf{piOEval}(\mathsf{C}_{\mathsf{Add,piO}}, ((\mathsf{ct}_{1,0}, \mathsf{ct}_{1,1}, \mathbf{i}, \pi_1), (\mathsf{ct}_{2,0}, \mathsf{ct}_{2,1}, \mathbf{i}, \pi_2), y)).$$

- $\mathsf{Inv}(\mathsf{pp}, [a]_{[\mathbf{i}]})$: On input the public parameter pp and an encoding $[[a]_{[\mathbf{i}]} = (\mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, \pi)$, output the encoding $[-a]_{[\mathbf{i}]} = (\mathsf{ct}^*_0, \mathsf{ct}^*_1, \mathbf{i}, \pi^*)$, where

$$(\mathsf{ct}^*_0, \mathsf{ct}^*_1, \mathbf{i}, \pi^*) = \mathsf{piOEval}(\mathsf{C}_{\mathsf{Inv,piO}}, (\mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, \pi, y)).$$

- $\mathsf{Mult}(\mathsf{pp}, [a_1]_{[\mathbf{i}]}, [a_2]_{[\mathbf{i}_2]})$: On input the public parameter pp and a pair of encodings

$$[a_1]_{[\mathbf{i}]} = (\mathsf{ct}_{1,0}, \mathsf{ct}_{1,1}, \mathbf{i}, \pi_1), \quad , \quad [a_2]_{[\mathbf{i}_2]} = (\mathsf{ct}_{2,0}, \mathsf{ct}_{2,1}, \mathbf{i}_2, \pi_2),$$

  output the encoding $[a_1.a_2]_{[\mathbf{i}_1+\mathbf{i}_2]} = (\mathsf{ct}^*_0, \mathsf{ct}^*_1, \mathbf{i}_1 + \mathbf{i}_2, \pi^*)$, where

$$(\mathsf{ct}^*_0, \mathsf{ct}^*_1, \mathbf{i}_1 + \mathbf{i}_2, \pi^*) = \mathsf{piOEval}(\mathsf{C}_{\mathsf{Mult,MMap}}, ((\mathsf{ct}_{1,0}, \mathsf{ct}_{1,1}, \mathbf{i}, \pi_1), (\mathsf{ct}_{2,0}, \mathsf{ct}_{2,1}, \mathbf{i}_2, \pi_2), y)).$$

- $\mathsf{Ext}(\mathsf{pp}, [a]_{[\mathbf{i}]})$: On input the public parameter pp and an encoding $[a]_{[\mathbf{i}]} = (\mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, \pi)$, output $g^* = \mathsf{piOEval}(\mathsf{C}_{\mathsf{Ext,piO}}, ((\mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, \pi), y))$.
- $\mathsf{ZeroTest}(\mathsf{pp}, [a]_{[\mathbf{i}]})$: On input the public parameter pp and an encoding $[a]_{[\mathbf{i}]} = (\mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, \pi)$, output b^*, where

$$b^* = \begin{cases} \perp & \text{if } \mathsf{piOEval}(\mathsf{C}_{\mathsf{Ext,piO}}, ((\mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, \pi), y)) = \perp, \\ 1 & \text{if } \mathsf{piOEval}(\mathsf{C}_{\mathsf{Ext,piO}}, ((\mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, \pi), y)) = g^0 \text{ for any group element } g \in \mathbb{G}, \\ 0 & \text{otherwise.} \end{cases}$$

4.7 Proof of SXDH

We state and prove the following theorem:

Theorem 4.1. *Our MMap construction is SXDH-hard assuming: (a) that deciding subset membership with respect to $\mathcal{L} \subset \mathcal{X}$ is computationally hard, (b) that NIZK satisfies* crs*-indistinguishability, perfect simulation-extractability under a binding* crs *and perfect zero-knowledge under a hiding* crs*, (c) that piO satisfies probabilistic-iO security, (d) that FHE is IND-CPA secure, and (e) that the group $\mathbb{G}$ is DDH-hard.*

We prove hardness of SXDH over our proposed MMap construction via a sequence of hybrids, summarized in Table 1. The crux of the proof lies in switching the plaintexts underlying the challenge encodings in the SXDH game to a special form such that:

- The challenge encodings constitute a valid SXDH instance over the MMap when the extraction circuit embeds a valid DDH tuple over the group $\mathbb{G}$.
- The challenge encodings constitute a random invalid SXDH instance over the MMap when the extraction circuit embeds a random invalid DDH tuple over the group $\mathbb{G}$.

We then show that these two cases are indistinguishable based on a sequence of hybrids, where each pair of consecutive hybrids is indistinguishable based on the computational security properties of the underlying cryptoprimitives and the hardness of DDH over the group $\mathbb{G}$.

Table 1: Overview of hybrids for proof of SXDH

Hybrid	crs	y	$\mathsf{C}_{\phi,\mathsf{MMap}}$ knows for $\phi \in \{\mathsf{Add}, \mathsf{Inv}, \mathsf{Mult}\}$	$\mathsf{C}_{\mathsf{Ext},\mathsf{MMap}}$ knows	(g, h, g', h') is of the form	Challenge FHE Ciphertexts (Set-1) are encryptions of	Challenge FHE Ciphertexts (Set-2) are encryptions of	Comments
0	binding	$\notin \mathcal{L}$	$(\mathsf{sk}_0, \mathsf{sk}_1, \mathsf{t})$	$(\mathsf{sk}_0, \mathsf{sk}_1, \mathsf{t})$	$(g, g^{\gamma_0}, g^{\gamma_1}, g^{\gamma^*})$	$(\alpha_0, 0, 0, 0)$ $(\alpha_1, 0, 0, 0)$ $(\alpha_0 \cdot \alpha_1, 0, 0, 0)$	$(\alpha_0, 0, 0, 0)$ $(\alpha_1, 0, 0, 0)$ $(\alpha_0 \cdot \alpha_1, 0, 0, 0)$	—
1	hiding	$\in \mathcal{L}$	—	sk_1	$(g, g^{\gamma_0}, g^{\gamma_1}, g^{\gamma_0 \cdot \gamma_1})$	$(\alpha_0, 0, 0, 0)$ $(\alpha_1, 0, 0, 0)$ $(\alpha_0 \cdot \alpha_1, 0, 0, 0)$	$(\alpha_0, 0, 0, 0)$ $(\alpha_1, 0, 0, 0)$ $(\alpha_0 \cdot \alpha_1, 0, 0, 0)$	NIZK security + subspace-membership hardness + piO security
2	hiding	$\in \mathcal{L}$	—	sk_1	$(g, g^{\gamma_0}, g^{\gamma_1}, g^{\gamma_0 \cdot \gamma_1})$	$((\alpha_0 - \mu_0 \cdot \gamma_0), \mu_0, 0, 0)$ $((\alpha_1 - \mu_1 \cdot \gamma_1), 0, \mu_1, 0, 0)$ $((\alpha_0 - \mu_0 \cdot \gamma_0) \cdot (\alpha_1 - \mu_1 \cdot \gamma_1), (\alpha_1 \cdot \mu_0), (\alpha_0 \cdot \mu_1), -(\mu_0 \cdot \mu_1))$	$(\alpha_0, 0, 0, 0)$ $(\alpha_1, 0, 0, 0)$ $(\alpha_0 \cdot \alpha_1, 0, 0, 0)$	FHE IND-CPA security
3	hiding	$\in \mathcal{L}$	—	sk_0	$(g, g^{\gamma_0}, g^{\gamma_1}, g^{\gamma_0 \cdot \gamma_1})$	$((\alpha_0 - \mu_0 \cdot \gamma_0), \mu_0, 0, 0)$ $((\alpha_1 - \mu_1 \cdot \gamma_1), 0, \mu_1, 0, 0)$ $((\alpha_0 - \mu_0 \cdot \gamma_0) \cdot (\alpha_1 - \mu_1 \cdot \gamma_1), (\alpha_1 \cdot \mu_0), (\alpha_0 \cdot \mu_1), -(\mu_0 \cdot \mu_1))$	$(\alpha_0, 0, 0, 0)$ $(\alpha_1, 0, 0, 0)$ $(\alpha_0 \cdot \alpha_1, 0, 0, 0)$	NIZK security + piO security
4	hiding	$\in \mathcal{L}$	—	sk_0	$(g, g^{\gamma_0}, g^{\gamma_1}, g^{\gamma_0 \cdot \gamma_1})$	$((\alpha_0 - \mu_0 \cdot \gamma_0), \mu_0, 0, 0)$ $((\alpha_1 - \mu_1 \cdot \gamma_1), 0, \mu_1, 0, 0)$ $((\alpha_0 - \mu_0 \cdot \gamma_0) \cdot (\alpha_1 - \mu_1 \cdot \gamma_1), (\alpha_1 \cdot \mu_0), (\alpha_0 \cdot \mu_1), -(\mu_0 \cdot \mu_1))$	$((\alpha_0 - \mu_0 \cdot \gamma_0), \mu_0, 0, 0)$ $((\alpha_1 - \mu_1 \cdot \gamma_1), 0, \mu_1, 0, 0)$ $((\alpha_0 - \mu_0 \cdot \gamma_0) \cdot (\alpha_1 - \mu_1 \cdot \gamma_1), (\alpha_1 \cdot \mu_0), (\alpha_0 \cdot \mu_1), -(\mu_0 \cdot \mu_1))$	FHE IND-CPA security
5	hiding	$\in \mathcal{L}$	—	sk_0	$(g, g^{\gamma_0}, g^{\gamma_1}, g^{\gamma^*})$	$((\alpha_0 - \mu_0 \cdot \gamma_0), \mu_0, 0, 0)$ $((\alpha_1 - \mu_1 \cdot \gamma_1), 0, \mu_1, 0, 0)$ $((\alpha_0 - \mu_0 \cdot \gamma_0) \cdot (\alpha_1 - \mu_1 \cdot \gamma_1), (\alpha_1 \cdot \mu_0), (\alpha_0 \cdot \mu_1), -(\mu_0 \cdot \mu_1))$	$((\alpha_0 - \mu_0 \cdot \gamma_0), \mu_0, 0, 0)$ $((\alpha_1 - \mu_1 \cdot \gamma_1), 0, \mu_1, 0, 0)$ $((\alpha_0 - \mu_0 \cdot \gamma_0) \cdot (\alpha_1 - \mu_1 \cdot \gamma_1), (\alpha_1 \cdot \mu_0), (\alpha_0 \cdot \mu_1), -(\mu_0 \cdot \mu_1))$	DDH
6	hiding	$\in \mathcal{L}$	—	sk_0	$(g, g^{\gamma_0}, g^{\gamma_1}, g^{\gamma^*})$	$((\alpha_0 - \mu_0 \cdot \gamma_0), \mu_0, 0, 0)$ $((\alpha_1 - \mu_1 \cdot \gamma_1), 0, \mu_1, 0, 0)$ $((\alpha_0 - \mu_0 \cdot \gamma_0) \cdot (\alpha_1 - \mu_1 \cdot \gamma_1), (\alpha_1 \cdot \mu_0), (\alpha_0 \cdot \mu_1), -(\mu_0 \cdot \mu_1))$	$(\alpha_0, 0, 0, 0)$ $(\alpha_1, 0, 0, 0)$ $((\alpha_0 \cdot \alpha_1 + \mu_0 \cdot \mu_1 \cdot (\gamma_0 \cdot \gamma_1 - \gamma^*)), 0, 0, 0)$	FHE IND-CPA security
7	hiding	$\in \mathcal{L}$	—	sk_1	$(g, g^{\gamma_0}, g^{\gamma_1}, g^{\gamma^*})$	$((\alpha_0 - \mu_0 \cdot \gamma_0), \mu_0, 0, 0)$ $((\alpha_1 - \mu_1 \cdot \gamma_1), 0, \mu_1, 0, 0)$ $((\alpha_0 - \mu_0 \cdot \gamma_0) \cdot (\alpha_1 - \mu_1 \cdot \gamma_1), (\alpha_1 \cdot \mu_0), (\alpha_0 \cdot \mu_1), -(\mu_0 \cdot \mu_1))$	$(\alpha_0, 0, 0, 0)$ $(\alpha_1, 0, 0, 0)$ $((\alpha_0 \cdot \alpha_1 + \mu_0 \cdot \mu_1 \cdot (\gamma_0 \cdot \gamma_1 - \gamma^*)), 0, 0, 0)$	NIZK security + piO security
8	hiding	$\in \mathcal{L}$	—	sk_1	$(g, g^{\gamma_0}, g^{\gamma_1}, g^{\gamma^*})$	$(\alpha_0, 0, 0, 0)$ $(\alpha_1, 0, 0, 0)$ $((\alpha_0 \cdot \alpha_1 + \mu_0 \cdot \mu_1 \cdot (\gamma_0 \cdot \gamma_1 - \gamma^*)), 0, 0, 0)$	$(\alpha_0, 0, 0, 0)$ $(\alpha_1, 0, 0, 0)$ $((\alpha_0 \cdot \alpha_1 + \mu_0 \cdot \mu_1 \cdot (\gamma_0 \cdot \gamma_1 - \gamma^*)), 0, 0, 0)$	FHE IND-CPA security
9	binding	$\notin \mathcal{L}$	$(g, g^{\gamma_0}, g^{\gamma_1}, g^{\gamma^*})$	$(\mathsf{sk}_0, \mathsf{sk}_1, \mathsf{t})$	$(\mathsf{sk}_0, \mathsf{sk}_1, \mathsf{t})$	$(\alpha_0, 0, 0, 0)$ $(\alpha_1, 0, 0, 0)$ $((\alpha_0 \cdot \alpha_1 + \mu_0 \cdot \mu_1 \cdot (\gamma_0 \cdot \gamma_1 - \gamma^*)), 0, 0, 0)$	$(\alpha_0, 0, 0, 0)$ $(\alpha_1, 0, 0, 0)$ $((\alpha_0 \cdot \alpha_1 + \mu_0 \cdot \mu_1 \cdot (\gamma_0 \cdot \gamma_1 - \gamma^*)), 0, 0, 0)$	NIZK security + subspace-membership hardness + piO security

4.7.1 Description of the Hybrids

We begin by describing the proof hybrids in details.

Hybrid-0. In this hybrid, the public parameter pp is generated as in the real scheme. As part of the SXDH challenge, the adversary is provided with encodings of α_0, α_1 (uniformly random in $\mathbb{Z}_q$) and $\alpha_0 \cdot \alpha_1$, generated as per the real encoding algorithm corresponding to some challenge level $\mathbf{i}$ (may be adversarially chosen).

Hybrid-1. In this hybrid, we change the public parameter pp as follows:

1. Switch crs for the NIZK proof system to hiding mode from binding mode.
2. Switch $y \in \mathcal{X}$ from a uniform non-member instance to a uniform member instance for the language $\mathcal{L}$ with witness wit_y.
3. Compute $\mathsf{C}_{\mathsf{Add},\mathsf{piO}} = \mathsf{piO.Obf}(\widehat{\mathsf{C}}_{\mathsf{Add},\mathsf{MMap}})$, where $\widehat{\mathsf{C}}_{\mathsf{Add},\mathsf{MMap}}$ is as described in Figure 9.
4. Compute $\mathsf{C}_{\mathsf{Inv},\mathsf{piO}} = \mathsf{piO.Obf}(\widehat{\mathsf{C}}_{\mathsf{Inv},\mathsf{MMap}})$, where $\widehat{\mathsf{C}}_{\mathsf{Inv},\mathsf{MMap}}$ is as described in Figure 10.
5. Compute $\mathsf{C}_{\mathsf{Mult},\mathsf{piO}} = \mathsf{piO.Obf}(\widehat{\mathsf{C}}_{\mathsf{Mult},\mathsf{MMap}})$, where $\widehat{\mathsf{C}}_{\mathsf{Mult},\mathsf{MMap}}$ is as described in Figure 11.
6. Compute $\mathsf{C}_{\mathsf{Ext},\mathsf{piO}} = \mathsf{piO.Obf}(\widehat{\mathsf{C}}_{\mathsf{Ext},\mathsf{MMap},1,0})$, where $\widehat{\mathsf{C}}_{\mathsf{Ext},\mathsf{MMap},1,0}$ is as described in Figure 12.

Inputs: A tuple of the form: $((\mathsf{ct}_{1,0}, \mathsf{ct}_{1,1}, \mathbf{i}, \pi_1), (\mathsf{ct}_{2,0}, \mathsf{ct}_{2,1}, \mathbf{i}, \pi_2), y \in \mathcal{X})$, where the proofs π_1 and π_2 are over the language L.

Outputs: Either a tuple of the form $(\mathsf{ct}_0^*, \mathsf{ct}_1^*, \mathbf{i}, \pi^*)$, or $\perp$.

Computation: Proceed as follows:

1. **If** any of the following is true:
   (a) $\mathsf{NIZK.Verify}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_{1,0}, \mathsf{ct}_{1,1}, \mathbf{i}, y \in \mathcal{X}), \pi_1) = 0$.
   (b) $\mathsf{NIZK.Verify}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_{2,0}, \mathsf{ct}_{2,1}, \mathbf{i}, y \in \mathcal{X}), \pi_2) = 0$.
   then output $\perp$.
2. // Decryption omitted — avoids dependency on sk_0 and sk_1.
3. Compute the following:
$$\mathsf{ct}_0^* = \mathsf{FHE.Eval}(\mathsf{pk}_0, \mathsf{ct}_{1,0}, \mathsf{ct}_{2,0}, \mathsf{C}_{\mathsf{Add},\mathsf{FHE}}) \quad , \quad \mathsf{ct}_1^* = \mathsf{FHE.Eval}(\mathsf{pk}_1, \mathsf{ct}_{1,1}, \mathsf{ct}_{2,1}, \mathsf{C}_{\mathsf{Add},\mathsf{FHE}}).$$
4. // If check omitted — avoids dependency on sk_0 and sk_1.
   Proceed as follows:
   (a) // Extraction omitted — avoids dependency on extraction trapdoor t.
   (b) // Check omitted.
   (c) Generate $\pi^* \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0^*, \mathsf{ct}_1^*, \mathbf{i}, y), \mathsf{wit}_y)$.
5. // Proof generation omitted — avoids dependency on sk_0 and sk_1.
6. Output $(\mathsf{ct}_0^*, \mathsf{ct}_1^*, \mathbf{i}, \pi^*)$.

Figure 9: Let $\mathsf{crs} \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$ be in hiding mode. Additionally, let $(\mathsf{pk}_0, \mathsf{sk}_0), (\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathsf{FHE.Gen}(1^\lambda)$. Also, for $y \in \mathcal{L}$, let wit_y be a membership-witness such that $(y, \mathsf{wit}_y) \in \mathsf{R}_{\mathcal{L}}$. We define a probabilistic circuit $\widehat{\mathsf{C}}_{\mathsf{Add},\mathsf{MMap}}$ (which has only the public keys pk_0 and pk_1 hard-wired into it) as above.

Hybrid-2. In this hybrid, we change the challenge encodings. Suppose that the DDH instance (g, h, g', h') in the extraction circuit $\widehat{C}_{\mathsf{Ext},\mathsf{MMap},1,0}$ is distributed as:
$$(g, h, g', h') = (g, g^{\gamma_0}, g^{\gamma_1}, g^{\gamma_0 \cdot \gamma_1}).$$
We switch the plaintext underlying the *first* FHE ciphertext in each challenge encoding as follows (where μ_0 and μ_1 are sampled uniformly from $\mathbb{Z}_q^*$):
$$\begin{aligned}
(\alpha_0, 0, 0, 0, \mathbf{i}) &\rightarrow ((\alpha_0 - \mu_0 \cdot \gamma_0), \mu_0, 0, 0, \mathbf{i}) \\
(\alpha_1, 0, 0, 0, \mathbf{i}) &\rightarrow ((\alpha_1 - \mu_1 \cdot \gamma_1), 0, \mu_1, 0, 0, \mathbf{i}) \\
(\alpha_0 \cdot \alpha_1, 0, 0, 0, \mathbf{i}) &\rightarrow ((\alpha_0 - \mu_0 \cdot \gamma_0) \cdot (\alpha_1 - \mu_1 \cdot \gamma_1), (\alpha_1 \cdot \mu_0), (\alpha_0 \cdot \mu_1), -(\mu_0 \cdot \mu_1), \mathbf{i})
\end{aligned}$$

Inputs: A tuple of the form: $(\{\mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, \pi, y \in \mathcal{X})$, where the proof π is over the language L.

Outputs: Either a tuple of the form $(\mathsf{ct}_0^*, \mathsf{ct}_1^*, \mathbf{i}, \pi^*)$, or $\perp$.

Computation: Proceed as follows:

1. If $\mathsf{NIZK.Verify}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, y), \pi) = 0$, then outut $\perp$.
2. // Decryption omitted — avoids dependency on sk_0 and sk_1.
3. Compute the following:
$$\mathsf{ct}_0^* = \mathsf{FHE.Eval}(\mathsf{pk}_0, \mathsf{ct}_0, \mathsf{C}_{\mathsf{Inv,FHE}}) \quad , \quad \mathsf{ct}_1^* = \mathsf{FHE.Eval}(\mathsf{pk}_1, \mathsf{ct}_1, \mathsf{C}_{\mathsf{Inv,FHE}}).$$
4. // If check omitted — avoids dependency on sk_0 and sk_1.
Proceed as follows:
   (a) // Extraction omitted — avoids dependency on extraction trapdoor t.
   (b) // Check omitted.
   (c) Generate $\pi^* \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0^*, \mathsf{ct}_1^*, \mathbf{i}, y), \mathsf{wit}_y)$.
5. // Proof generation omitted — avoids dependency on sk_0 and sk_1.
6. Output $(\mathsf{ct}_0^*, \mathsf{ct}_1^*, \mathbf{i}, \pi^*)$.

Figure 10: Next, Let $\mathsf{crs} \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$ be in hiding mode. Additionally, let $(\mathsf{pk}_0, \mathsf{sk}_0), (\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathsf{FHE.Gen}(1^\lambda)$. Also, for $y \in \mathcal{L}$, let wit_y be a membership-witness such that $(y, \mathsf{wit}_y) \in \mathsf{R}_{\mathcal{L}}$. We define a probabilistic circuit $\widehat{\mathsf{C}}_{\mathsf{Inv,MMap}}$ (which has only the public keys pk_0 and pk_1 hard-wired into it) as above

Inputs: A tuple of the form: $((\mathsf{ct}_{1,0}, \mathsf{ct}_{1,1}, \mathbf{i}_1, \pi_1), (\mathsf{ct}_{2,0}, \mathsf{ct}_{2,1}, \mathbf{i}_2, \pi_2), y \in \mathcal{X})$, where the proofs π_1 and π_2 are over the language L.

Outputs: Either a tuple of the form $(\mathsf{ct}_0^*, \mathsf{ct}_1^*, i, k, \pi^*)$, or $\perp$.

Computation: Proceed as follows:

1. **If** any of the following is true:
   (a) $\mathsf{NIZK.Verify}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_{1,0}, \mathsf{ct}_{1,1}, \mathbf{i}_1, y \in \mathcal{X}), \pi_1) = 0$.
   (b) $\mathsf{NIZK.Verify}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_{2,0}, \mathsf{ct}_{2,1}, \mathbf{i}_2, y \in \mathcal{X}), \pi_2) = 0$.
   then output $\perp$.
2. // Decryption omitted — avoids dependency on sk_0 and sk_1.
3. Compute the following:
$$\mathsf{ct}_0^* = \mathsf{FHE.Eval}(\mathsf{pk}_0, \mathsf{ct}_{1,0}, \mathsf{ct}_{2,0}, \mathsf{C}_{\mathsf{Mult,FHE}}) \quad , \quad \mathsf{ct}_1^* = \mathsf{FHE.Eval}(\mathsf{pk}_1, \mathsf{ct}_{1,1}, \mathsf{ct}_{2,1}, \mathsf{C}_{\mathsf{Mult,FHE}}).$$
4. // If check omitted — avoids dependency on sk_0 and sk_1.
Proceed as follows:
   (a) // Extraction omitted — avoids dependency on extraction trapdoor t.
   (b) // Check omitted.
   (c) Generate $\pi^* \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0^*, \mathsf{ct}_1^*, \mathbf{i}_1 + \mathbf{i}_2, y), \mathsf{wit}_y)$.
5. // Proof generation omitted — avoids dependency on sk_0 and sk_1.
6. Output $(\mathsf{ct}_0^*, \mathsf{ct}_1^*, \mathbf{i}_1 + \mathbf{i}_2, \pi^*)$.

Figure 11: Let $\mathsf{crs} \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$ be in hiding mode. Additionally, let $(\mathsf{pk}_0, \mathsf{sk}_0), (\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathsf{FHE.Gen}(1^\lambda)$. Also, for $y \in \mathcal{L}$, let wit_y be a membership-witness such that $(y, \mathsf{wit}_y) \in \mathsf{R}_{\mathcal{L}}$. For each $b^* \in \{0, 1\}$, we define a probabilistic circuit $\widehat{\mathsf{C}}_{\mathsf{Mult,MMap}}$ (which has only the public keys pk_0 and pk_1 hard-wired into it) as above.

Hybrid-3. In this hybrid, we change the public parameter pp as follows: compute $\mathsf{C}_{\mathsf{Ext,piO}} = \mathsf{piO.Obf}(\widehat{\mathsf{C}}_{\mathsf{Ext,MMap},0,0})$, where $\widehat{\mathsf{C}}_{\mathsf{Ext,MMap},0,0}$ is as defined in Figure 12.

Inputs: A tuple of the form: $((\mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, \pi), y \in \mathcal{X})$, where the proof π is over the language L.

Outputs: Either a group element g^* or $\perp$.

Computation: Proceed as follows:

1. If $\mathsf{NIZK.Verify}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, y), \pi) = 0$, then output $\perp$.
2. Recover the following: $(\{a_{\beta,\ell}\}_{\ell \in [0,3]}, \mathbf{i}_\beta) = \mathsf{FHE.Dec}(\mathsf{sk}_\beta, \mathsf{ct}_\beta)$.
   // Decryption using $\mathsf{sk}_{1-\beta}$ omitted.
3. Compute $g^* = g^{a_{\beta,0}} \cdot h^{a_{\beta,1}} \cdot (g')^{a_{\beta,2}} \cdot (h')^{a_{\beta,3}}$.
4. // If clause omitted — avoids dependency on $\mathsf{sk}_{1-\beta}$ and extraction trapdoor t.
5. Output g^*.

Figure 12: Let $\mathsf{crs} \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$ be in hiding mode and let $(\mathsf{pk}_0, \mathsf{sk}_0), (\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathsf{FHE.Gen}(1^\lambda)$. Also, let $g, h, g', h' \in \mathbb{G}$ be group elements that are distributed as per $\mathcal{D}_{\mathrm{DDH},\mathbb{G}}$ if $\beta' = 0$ and uniformly randomly sampled if $\beta' = 1$, subject to the restriction that the tuple always uses the same base element g. For each $\beta, \beta' \in \{0,1\}$, we define a deterministic circuit $\widehat{C}_{\mathsf{Ext,MMap},\beta,\beta'}$ (which has the public keys pk_0 and pk_1, the secret key sk_β and the collection of group elements (g, h, g', h') hard-wired into it) as in Figure 12 above.

Hybrid-4. In this hybrid, we change the challenge encodings. Again, suppose that the DDH instance (g, h, g', h') in the extraction circuit $\widehat{C}_{\mathsf{Ext,MMap},0,0}$ is distributed as:

$$(g, h, g', h') = (g, g^{\gamma_0}, g^{\gamma_1}, g^{\gamma_0 \cdot \gamma_1}).$$

Switch the plaintext underlying the *second* FHE ciphertext in each challenge encoding as follows (where μ_0 and μ_1 are sampled uniformly from $\mathbb{Z}_q^*$):

$$\begin{aligned}
(\alpha_0, 0, 0, 0, \mathbf{i}) &\rightarrow ((\alpha_0 - \mu_0 \cdot \gamma_0), \mu_0, 0, 0, \mathbf{i}) \\
(\alpha_1, 0, 0, 0, \mathbf{i}) &\rightarrow ((\alpha_1 - \mu_1 \cdot \gamma_1), 0, \mu_1, 0, 0, \mathbf{i}) \\
(\alpha_0 \cdot \alpha_1, 0, 0, 0, \mathbf{i}) &\rightarrow ((\alpha_0 - \mu_0 \cdot \gamma_0) \cdot (\alpha_1 - \mu_1 \cdot \gamma_1), (\alpha_1 \cdot \mu_0), (\alpha_0 \cdot \mu_1), -(\mu_0 \cdot \mu_1), \mathbf{i})
\end{aligned}$$

Hybrid-5. In this hybrid, we change the public parameter pp as follows: compute $\mathsf{C}_{\mathsf{Ext,piO}} = \mathsf{piO.Obf}(\widehat{\mathsf{C}}_{\mathsf{Ext,MMap},0,1})$, where $\widehat{\mathsf{C}}_{\mathsf{Ext,MMap},0,1}$ is as defined in Figure 12. Note that the only difference between the extraction circuits $\widehat{\mathsf{C}}_{\mathsf{Ext,MMap},0,0}$ and $\widehat{\mathsf{C}}_{\mathsf{Ext,MMap},0,1}$ is that the former embeds a valid DDH tuple over the group $\mathbb{G}$ while the latter embeds a uniformly random four-tuple of group elements over the group $\mathbb{G}$.

Hybrid-6. In this hybrid, we change the challenge encodings. Recall that in hybrid-5, (g, h, g', h') in the extraction circuit $\widehat{C}_{\mathsf{Ext,MMap},0,1}$ is distributed as:

$$(g, h, g', h') = (g, g^{\gamma_0}, g^{\gamma_1}, g^{\gamma^*}).$$

Switch back the plaintext underlying the *second* FHE ciphertext in each challenge encoding as follows (where μ_0 and μ_1 are sampled uniformly from $\mathbb{Z}_q^*$):

$$\begin{aligned}
((\alpha_0 - \mu_0 \cdot \gamma_0), \mu_0, 0, 0, \mathbf{i}) &\rightarrow (\alpha_0, 0, 0, 0, \mathbf{i}) \\
((\alpha_1 - \mu_1 \cdot \gamma_1), 0, \mu_1, 0, 0, \mathbf{i}) &\rightarrow (\alpha_1, 0, 0, 0, \mathbf{i}) \\
((\alpha_0 - \mu_0 \cdot \gamma_0) \cdot (\alpha_1 - \mu_1 \cdot \gamma_1), (\alpha_1 \cdot \mu_0), (\alpha_0 \cdot \mu_1), -(\mu_0 \cdot \mu_1), \mathbf{i}) &\rightarrow ((\alpha_0 \cdot \alpha_1 + \mu_0 \cdot \mu_1 \cdot (\gamma_0 \cdot \gamma_1 - \gamma^*)), 0, 0, 0, \mathbf{i})
\end{aligned}$$

Hybrid-7. In this hybrid, we change the public parameter pp as follows: compute $\mathsf{C}_{\mathsf{Ext,piO}} = \mathsf{piO.Obf}(\widehat{\mathsf{C}}_{\mathsf{Ext,MMap},1,1})$, where $\widehat{\mathsf{C}}_{\mathsf{Ext,MMap},1,1}$ is as defined in Figure 12.

Hybrid-8. In this hybrid, we change the challenge encodings. Recall that in hybrid-7, (g, h, g', h') in the extraction circuit $\widehat{C}_{\mathsf{Ext,MMap},1,1}$ is distributed as:

$$(g, h, g', h') = (g, g^{\gamma_0}, g^{\gamma_1}, g^{\gamma^*}).$$

Switch back the plaintext underlying the *first* FHE ciphertext in each challenge encoding as follows (where μ_0 and μ_1 are sampled uniformly from $\mathbb{Z}_q^*$):

$$\begin{aligned}((\alpha_0 - \mu_0 \cdot \gamma_0), \mu_0, 0, 0, \mathbf{i}) &\rightarrow (\alpha_0, 0, 0, 0, \mathbf{i})\\ ((\alpha_1 - \mu_1 \cdot \gamma_1), 0, \mu_1, 0, 0, \mathbf{i}) &\rightarrow (\alpha_1, 0, 0, 0, \mathbf{i})\\ ((\alpha_0 - \mu_0 \cdot \gamma_0) \cdot (\alpha_1 - \mu_1 \cdot \gamma_1), (\alpha_1 \cdot \mu_0), (\alpha_0 \cdot \mu_1), -(\mu_0 \cdot \mu_1), \mathbf{i}) &\rightarrow ((\alpha_0 \cdot \alpha_1 + \mu_0 \cdot \mu_1 \cdot (\gamma_0 \cdot \gamma_1 - \gamma^*)), 0, 0, 0, \mathbf{i})\end{aligned}$$

Hybrid-9. In this hybrid, we change the public parameter pp as follows:

1. Switch crs for the NIZK proof system to binding mode, as in the real scheme
2. Switch $y \in \mathcal{X}$ to a uniform non-member instance, as in the real scheme.
3. Compute $\mathsf{C}_{\mathsf{Add,piO}} = \mathsf{piO.Obf}(\mathsf{C}_{\mathsf{Add,MMap}})$, as in the real scheme.
4. Compute $\mathsf{C}_{\mathsf{Inv,piO}} = \mathsf{piO.Obf}(\mathsf{C}_{\mathsf{Inv,MMap}})$, as in the real scheme.
5. Compute $\mathsf{C}_{\mathsf{Mult,piO}} = \mathsf{piO.Obf}(\mathsf{C}_{\mathsf{Mult,MMap}})$, as in the real scheme.
6. Compute $\mathsf{C}_{\mathsf{Ext,piO}} = \mathsf{piO.Obf}(\mathsf{C}_{\mathsf{Ext,MMap}})$, as in the real scheme.

Finally, Hybrid-9 is identical to the game where the adversary is given a random invalid SXDH instance generated as per the real scheme.

4.7.2 Indistinguishability of the Hybrids

We now formally prove that each pair of consecutive hybrids is computationally indistinguishable from each other.

Lemma 4.2. *Hybrid-1 is computationally indistinguishable from Hybrid-0 assuming: (a) that deciding subset membership with respect to $\mathcal{L} \subset \mathcal{X}$ is computationally hard, (b) that NIZK satisfies* crs*-indistinguishability, perfect simulation-extractability under a binding* crs *and perfect zero-knowledge under a hiding* crs*, and (c) that piO satisfies probabilistic-iO security.*

Proof. We prove this via a sequence of four sub-hybrids:

1. In the **first sub-hybrid**, we switch $y \in \mathcal{X}$ from a uniform non-member instance to a uniform member instance for the language $\mathcal{L}$ with witness wit_y. This is obviously indistinguishable from the Hybrid-0 based on the hardness of deciding membership in $\mathcal{L}$.

2. In the **second sub-hybrid**, we hardwire the witness wit_y into the addition circuit $\mathsf{C}_{\mathsf{Add,MMap}}$, the inversion circuit $\mathsf{C}_{\mathsf{Inv,MMap}}$, the multiplication circuit $\mathsf{C}_{\mathsf{Mult,MMap}}$ and the extraction circuit $\mathsf{C}_{\mathsf{Ext,MMap}}$. We also remove the NIZK extraction trapdoor $\mathsf{t}_{\mathsf{ext}}$ from each of these circuits.

   We claim that this change does not change the functionality of $\mathsf{C}_{\mathsf{Add,MMap}}$, $\mathsf{C}_{\mathsf{Inv,MMap}}$, $\mathsf{C}_{\mathsf{Mult,MMap}}$ and $\mathsf{C}_{\mathsf{Ext,MMap}}$ at all. We remind the reader that $\mathcal{L}$ has unique witnesses membership witnesses. Hence, any witness wit'_y for $y \in \mathcal{L}$ extracted by the any of these circuits using the extraction trapdoor in Hybrid-0 must be equal to the hardwired witness wit_y in Hybrid-1. Since crs is binding, extraction always succeeds in Hybrid-0.

Hence, assuming that NIZK is perfectly simulation-extractable (under a binding crs) and that piO satisfies probabilistic-iO security, the second sub-hybrid is computationally indistinguishable from the first sub-hybrid.

3. In the **third sub-hybrid**, we switch the crs for the NIZK proof system to hiding mode from binding mode. This is again obviously indistinguishable from the second sub-hybrid based on the crs-indistinguishability property of NIZK.

4. In the **fourth sub-hybrid**, we switch $\mathsf{C}_{\mathsf{Add,MMap}}$, $\mathsf{C}_{\mathsf{Inv,MMap}}$, $\mathsf{C}_{\mathsf{Mult,MMap}}$ and $\mathsf{C}_{\mathsf{Ext,MMap}}$ to $\widehat{\mathsf{C}}_{\mathsf{Add,MMap}}$, $\widehat{\mathsf{C}}_{\mathsf{Inv,MMap}}$, $\widehat{\mathsf{C}}_{\mathsf{Mult,MMap}}$ and $\widehat{\mathsf{C}}_{\mathsf{Ext,MMap},1,0}$, respectively. The first three circuits now use neither sk_0 nor sk_1, while the transformed extraction circuit $\widehat{\mathsf{C}}_{\mathsf{Ext,MMap},1,0}$only uses the secret key sk_1.

   Once again, these modifications do not alter the output distributions of the aforementioned circuits. In particular, the only potential changes made are to the distributions of the proofs that use wit_y as witness. However, by the perfect zero-knowledge property of the proof system (under a hiding crs), the distributions of the resulting proofs remain identical.

   Hence, assuming that NIZK is perfectly zero-knowledge (under a hiding crs) and that piO satisfies probabilistic-iO security, the fourth sub-hybrid is computationally indistinguishable from the third sub-hybrid.

   Finally, it is easy to see that the fourth sub-hybrid is identical to Hybrid-1.

This completes the proof of computational indistinguishability of hybrids 0 and 1.

Lemma 4.3. *Hybrid-2 is computationally indistinguishable from Hybrid-1 assuming that FHE is IND-CPA secure with respect to the public key-secret key pair* $(\mathsf{pk}_0, \mathsf{sk}_0)$.

Proof. Consider a PPT distinguisher $\mathcal{B}_2$ against the IND-CPA security of FHE with respect to the key pair $(\mathsf{pk}_0, \mathsf{sk}_0)$. $\mathcal{B}_2$ simulates the challenger in Hybrid-1 and interacts with the SXDH adversary $\mathcal{A}$. More specifically, $\mathcal{B}_2$ sets up the public parameters for the MMap exactly as the challenger does in Hybrid-1. In particular, it samples an additional key pair $(\mathsf{pk}_1, \mathsf{sk}_1)$ for the FHE scheme, and uses it to generate the piO obfuscated circuits for addition, inversion, multiplication and extraction over encodings. Note that it does not have not access to sk_0, but this is not a hindrance as sk_0 is not embedded in any of the aforementioned circuits in Hybrid-1.

Next, when generating the challenge encodings, it creates the following sets of plaintext-pairs for the FHE scheme:

Set-0	**Set-1**
$(\alpha_0, 0, 0, 0, \mathbf{i})$	$((\alpha_0 - \mu_0 \cdot \gamma_0), \mu_0, 0, 0, \mathbf{i})$
$(\alpha_1, 0, 0, 0, \mathbf{i})$	$((\alpha_1 - \mu_1 \cdot \gamma_1), 0, \mu_1, 0, 0, \mathbf{i})$
$(\alpha_0 \cdot \alpha_1, 0, 0, 0, \mathbf{i})$	$((\alpha_0 - \mu_0 \cdot \gamma_0) \cdot (\alpha_1 - \mu_1 \cdot \gamma_1), (\alpha_1 \cdot \mu_0), (\alpha_0 \cdot \mu_1), -(\mu_0 \cdot \mu_1), \mathbf{i})$

where $\alpha_0, \alpha_1, \mu_0, \mu_1, \gamma_0$ and γ_1 are distributed exactly as in Hybrids 1 and 2. Next, $\mathcal{B}_2$ proceeds as follows:

1. For generating the first set of FHE ciphertexts in the challenge encodings, it forwards each pair of plaintexts as described above to the challenger in the FHE CPA-security experiment, and receives an encryption of the messages from Set-b for some unknown bit b under the key pair $(\mathsf{pk}_0, \mathsf{sk}_0)$.
2. For generating the second set of FHE ciphertexts in the challenge encodings, it encrypts the messages from Set-0 under the key pair $(\mathsf{pk}_1, \mathsf{sk}_1)$.

$\mathcal{B}_2$ then generates a *simulated* NIZK proof π for each challenge encoding using the simulation crs and simulation trapdoor $\mathsf{t}_{\mathcal{S}}$ of the NIZK proof system (guaranteed by the zero-knowledge property). Finally, it forwards to the adversary $\mathcal{A}$ the challenge encodings consisting of the FHE ciphertexts as created above and the simulated NIZK proofs.

Eventually, $\mathcal{A}$ outputs a bit b'. $\mathcal{B}_2$ outputs b' as a guess for the bit b used by the challenger in the FHE CPA-security experiment.

Note that when $b = 0$, $\mathcal{B}_2$ perfectly simulates Hybrid-1 and when $b = 1$, $\mathcal{B}_2$ perfectly simulates Hybrid-2. This completes the proof of indistinguishability of hybrids 1 and 2.

Lemma 4.4. *Hybrid-3 is computationally indistinguishable from Hybrid-2 assuming: (a) that NIZK satisfies* crs*-indistinguishability, perfect simulation-extractability under a binding* crs *and perfect zero-knowledge under a hiding* crs*, and (b) that piO satisfies probabilistic-iO security.*

Proof. We prove this via a sequence of three sub-hybrids:

1. In the first sub-hybrid, we switch the crs for the NIZK proof system to binding mode from hiding mode. This is obviously indistinguishable from Hybrid-2 based on the crs-indistinguishability property of NIZK.

2. In the second sub-hybrid, we switch the obfuscated circuit $\mathsf{C}_{\mathsf{Ext,piO}}$ in the public parameter from an obfuscation of $\widehat{\mathsf{C}}_{\mathsf{Ext,MMap},1,0}$ to an obfuscation of $\widehat{\mathsf{C}}_{\mathsf{Ext,MMap},0,0}$ (where both circuits are as defined in Figure 12).

3. In the third sub-hybrid, we switch back the crs for the NIZK proof system to hiding mode from binding mode. This is again obviously indistinguishable from the second sub-hybrid based on the crs-indistinguishability property of NIZK. Also, the third sub-hybrid is identical to Hybrid-3.

It remains to prove that the second sub-hybrid is computationally indistinguishable from the first sub-hybrid. We prove this below based on piO security.

Consider a PPT distinguisher $\mathcal{B}_3$ against the security of piO. $\mathcal{B}_3$ simulates the challenger in Hybrid-2 and interacts with the SXDH adversary $\mathcal{A}$. More specifically, $\mathcal{B}_3$ sets up the public parameters for the MMap exactly as the challenger does in Hybrid-2, except for the manner in which $\mathsf{C}_{\mathsf{Ext,piO}}$ is simulated as part of the public parameter.

Concretely, $\mathcal{B}_3$ forwards the fake extraction circuits $\widehat{\mathsf{C}}_{\mathsf{Ext,MMap},1,0}$ and $\widehat{\mathsf{C}}_{\mathsf{Ext,MMap},0,0}$ (where both circuits are as defined in Figure 12) to the challenger in the game against the security of piO, and receives an obfuscation of the circuit $\widehat{\mathsf{C}}_{\mathsf{Ext,MMap},b,0}$ for some unknown bit b. It uses this to simulate $\mathsf{C}_{\mathsf{Ext,piO}}$ in the public parameter.

Eventually $\mathcal{A}$ outputs a bit b'. $\mathcal{B}_3$ outputs b' as a guess for the bit b used by the challenger in the piO security experiment.

We note that $\widehat{\mathsf{C}}_{\mathsf{Ext,MMap},1,0}$ and $\widehat{\mathsf{C}}_{\mathsf{Ext,MMap},0,0}$ are functionally equivalent circuits with the same output distribution on all possible encodings. First of all, since the NIZK proof system is in binding mode and is perfectly simulation-extractable, and y is a uniform non-member, any adversarially generated encoding produces the same output when extracted from, irrespective of whether $\widehat{\mathsf{C}}_{\mathsf{Ext,MMap},1,0}$ or $\widehat{\mathsf{C}}_{\mathsf{Ext,MMap},0,0}$ is used. Finally, the extraction outputs on the challenge encodings (and any encodings derived from the challenge encodings) are, by design, equal for either extraction circuit.

Thus when $b = 0$, $\mathcal{B}_3$ perfectly simulates the first sub-hybrid and when $b = 1$, $\mathcal{B}_3$ perfectly simulates the second hybrid. This completes the proof of computational indistinguishability of the first and second sub-hybrids, and hence the overall indistinguishability of hybrids 2 and 3.

Lemma 4.5. *Hybrid-4 is computationally indistinguishable from Hybrid-3 assuming that FHE is IND-CPA secure with respect to the public key-secret key pair* $(\mathsf{pk}_1, \mathsf{sk}_1)$.

Proof. Consider a PPT distinguisher $\mathcal{B}_4$ against the IND-CPA security of FHE with respect to the key pair $(\mathsf{pk}_1, \mathsf{sk}_1)$. $\mathcal{B}_4$ simulates the challenger in Hybrid-3 and interacts with the SXDH adversary $\mathcal{A}$. More specifically, $\mathcal{B}_4$ sets up the public parameters for the MMap exactly as the challenger does in Hybrid-3. In particular, it samples an additional key pair $(\mathsf{pk}_0, \mathsf{sk}_0)$ for the FHE scheme, and uses it to generate the piO obfuscated circuits for addition, inversion, multiplication and extraction over encodings. Note that it does not have not access to sk_1, but this is not a hindrance as sk_1 is not embedded in any of the aforementioned circuits in Hybrid-3.

Next, when generating the challenge encodings, $\mathcal{B}_4$ creates the following sets of plaintext-pairs for the FHE scheme:

Set-0	**Set-1**
$(\alpha_0, 0, 0, 0, \mathbf{i})$	$((\alpha_0 - \mu_0 \cdot \gamma_0), \mu_0, 0, 0, \mathbf{i})$
$(\alpha_1, 0, 0, 0, \mathbf{i})$	$((\alpha_1 - \mu_1 \cdot \gamma_1), 0, \mu_1, 0, 0, \mathbf{i})$
$(\alpha_0 \cdot \alpha_1, 0, 0, 0, \mathbf{i})$	$((\alpha_0 - \mu_0 \cdot \gamma_0) \cdot (\alpha_1 - \mu_1 \cdot \gamma_1), (\alpha_1 \cdot \mu_0), (\alpha_0 \cdot \mu_1), -(\mu_0 \cdot \mu_1), \mathbf{i})$

where $\alpha_0, \alpha_1, \mu_0, \mu_1, \gamma_0$ and γ_1 are distributed exactly as in Hybrids 3 and 4. Next, $\mathcal{B}_4$ proceeds as follows:

1. For generating the first set of FHE ciphertexts in the challenge encodings, it encrypts the messages from Set-0 under the key pair $(\mathsf{pk}_0, \mathsf{sk}_0)$.
2. For generating the second set of FHE ciphertexts in the challenge encodings, it forwards each pair of plaintexts as described above to the challenger in the FHE CPA-security experiment, and receives an encryption of the messages from Set-b for some unknown bit b under the key pair $(\mathsf{pk}_1, \mathsf{sk}_1)$.

$\mathcal{B}_4$ then generates a *simulated* NIZK proof π for each challenge encoding using the simulation crs and simulation trapdoor $\mathsf{t}_{\mathcal{S}}$ of the NIZK proof system (guaranteed by the zero-knowledge property). Finally, it forwards to the adversary $\mathcal{A}$ the challenge encodings consisting of the FHE ciphertexts as created above and the simulated NIZK proofs.

Eventually, $\mathcal{A}$ outputs a bit b'. $\mathcal{B}_4$ outputs b' as a guess for the bit b used by the challenger in the FHE CPA-security experiment.

Note that when $b = 0$, $\mathcal{B}_4$ perfectly simulates Hybrid-3 and when $b = 1$, $\mathcal{B}_2$ perfectly simulates Hybrid-4. This completes the proof of indistinguishability of hybrids 3 and 4.

Lemma 4.6. *Hybrid-5 is computationally indistinguishable from Hybrid-4 assuming that the DDH assumption holds over the group* $\mathbb{G}$.

Proof. Note that the only difference between hybrids 4 and 5 is the manner in which the obfuscated extraction circuit $\mathsf{C}_{\mathsf{Ext,piO}}$ is generated as part of the public parameter.

In Hybrid-4, $\mathsf{C}_{\mathsf{Ext,piO}}$ is an obfuscation of the circuit $\widehat{\mathsf{C}}_{\mathsf{Ext,MMap},0,0}$, while in Hybrid-5, $\mathsf{C}_{\mathsf{Ext,piO}}$ is an obfuscation of the circuit $\widehat{\mathsf{C}}_{\mathsf{Ext,MMap},0,1}$ (where both circuits are as defined in Figure 12).

Now, observe that the only difference between the extraction circuits $\widehat{\mathsf{C}}_{\mathsf{Ext,MMap},0,0}$ and $\widehat{\mathsf{C}}_{\mathsf{Ext,MMap},0,1}$ is that the former embeds a valid DDH tuple over the group $\mathbb{G}$ while the latter embeds a uniformly random four-tuple of group elements over the group $\mathbb{G}$.

It immediately follows that Hybrid-5 is computationally indistinguishable from Hybrid-4 assuming that the DDH assumption holds over the group $\mathbb{G}$.

Lemma 4.7. *Hybrid-6 is computationally indistinguishable from Hybrid-5 assuming that FHE is IND-CPA secure with respect to the public key-secret key pair* $(\mathsf{pk}_1, \mathsf{sk}_1)$.

Proof. Consider a PPT distinguisher $\mathcal{B}_6$ against the IND-CPA security of FHE with respect to the key pair $(\mathsf{pk}_1, \mathsf{sk}_1)$. $\mathcal{B}_6$ simulates the challenger in Hybrid-5 and interacts with the SXDH adversary $\mathcal{A}$. More specifically, $\mathcal{B}_6$ sets up the public parameters for the MMap exactly as the challenger does in Hybrid-5. In particular, it samples an additional key pair $(\mathsf{pk}_0, \mathsf{sk}_0)$ for the FHE scheme, and uses it to generate the piO obfuscated circuits for addition, inversion, multiplication and extraction over encodings. Note that it does not have not access to sk_1, but this is not a hindrance as sk_1 is not embedded in any of the aforementioned circuits in Hybrid-5.

Next, when generating the challenge encodings, $\mathcal{B}_6$ creates the following sets of plaintext-pairs for the FHE scheme:

Set-0	**Set-1**
$((\alpha_0 - \mu_0 \cdot \gamma_0), \mu_0, 0, 0, \mathbf{i})$	$(\alpha_0, 0, 0, 0, \mathbf{i})$
$((\alpha_1 - \mu_1 \cdot \gamma_1), 0, \mu_1, 0, 0, \mathbf{i})$	$(\alpha_1, 0, 0, 0, \mathbf{i})$
$((\alpha_0 - \mu_0 \cdot \gamma_0) \cdot (\alpha_1 - \mu_1 \cdot \gamma_1), (\alpha_1 \cdot \mu_0), (\alpha_0 \cdot \mu_1), -(\mu_0 \cdot \mu_1), \mathbf{i})$	$((\alpha_0 \cdot \alpha_1 + \mu_0 \cdot \mu_1 \cdot (\gamma_0 \cdot \gamma_1 - \gamma^*)), 0, 0, 0, \mathbf{i})$

where $\alpha_0, \alpha_1, \mu_0, \mu_1, \gamma_0, \gamma_1$ and γ^* are distributed exactly as in Hybrids 5 and 6. Next, $\mathcal{B}_6$ proceeds as follows:

1. For generating the first set of FHE ciphertexts in the challenge encodings, it encrypts the messages from Set-0 under the key pair $(\mathsf{pk}_0, \mathsf{sk}_0)$.
2. For generating the second set of FHE ciphertexts in the challenge encodings, it forwards each pair of plaintexts as described above to the challenger in the FHE CPA-security experiment, and receives an encryption of the messages from Set-b for some unknown bit b under the key pair $(\mathsf{pk}_1, \mathsf{sk}_1)$.

$\mathcal{B}_6$ then generates a *simulated* NIZK proof π for each challenge encoding using the simulation crs and simulation trapdoor $\mathsf{t}_{\mathcal{S}}$ of the NIZK proof system (guaranteed by the zero-knowledge property). Finally, it forwards to the adversary $\mathcal{A}$ the challenge encodings consisting of the FHE ciphertexts as created above and the simulated NIZK proofs.

Eventually, $\mathcal{A}$ outputs a bit b'. $\mathcal{B}_6$ outputs b' as a guess for the bit b used by the challenger in the FHE CPA-security experiment.

Note that when $b = 0$, $\mathcal{B}_6$ perfectly simulates Hybrid-5 and when $b = 1$, $\mathcal{B}_6$ perfectly simulates Hybrid-6. This completes the proof of indistinguishability of hybrids 5 and 6.

Lemma 4.8. *Hybrid-7 is computationally indistinguishable from Hybrid-6 assuming: (a) that NIZK satisfies* crs*-indistinguishability, perfect simulation-extractability under a binding* crs *and perfect zero-knowledge under a hiding* crs*, and (b) that piO satisfies probabilistic-iO security.*

Proof. We prove this via a sequence of three sub-hybrids:

1. In the first sub-hybrid, we switch the crs for the NIZK proof system to binding mode from hiding mode. This is obviously indistinguishable from Hybrid-6 based on the crs-indistinguishability property of NIZK.
2. In the second sub-hybrid, we switch the obfuscated circuit $\mathsf{C}_{\mathsf{Ext,piO}}$ in the public parameter from an obfuscation of $\widehat{\mathsf{C}}_{\mathsf{Ext,MMap},0,1}$ to an obfuscation of $\widehat{\mathsf{C}}_{\mathsf{Ext,MMap},1,1}$ (where both circuits are as defined in Figure 12).

3. In the third sub-hybrid, we switch back the crs for the NIZK proof system to hiding mode from binding mode. This is again obviously indistinguishable from the second sub-hybrid based on the crs-indistinguishability property of NIZK. Also, the third sub-hybrid is identical to Hybrid-7.

It remains to prove that the second sub-hybrid is computationally indistinguishable from the first sub-hybrid. We prove this below based on piO security.

Consider a PPT distinguisher $\mathcal{B}_7$ against the security of piO. $\mathcal{B}_7$ simulates the challenger in Hybrid-6 and interacts with the SXDH adversary $\mathcal{A}$. More specifically, $\mathcal{B}_7$ sets up the public parameters for the MMap exactly as the challenger does in Hybrid-6, except for the manner in which $\mathsf{C}_{\mathsf{Ext,piO}}$ is simulated as part of the public parameter.

Concretely, $\mathcal{B}_7$ forwards the fake extraction circuits $\widehat{\mathsf{C}}_{\mathsf{Ext,MMap},0,1}$ and $\widehat{\mathsf{C}}_{\mathsf{Ext,MMap},1,1}$ (where both circuits are as defined in Figure 12) to the challenger in the game against the security of piO, and receives an obfuscation of the circuit $\widehat{\mathsf{C}}_{\mathsf{Ext,MMap},b,1}$ for some unknown bit b. It uses this to simulate $\mathsf{C}_{\mathsf{Ext,piO}}$ in the public parameter.

Eventually $\mathcal{A}$ outputs a bit b'. $\mathcal{B}_3$ outputs b' as a guess for the bit b used by the challenger in the piO security experiment.

We note that $\widehat{\mathsf{C}}_{\mathsf{Ext,MMap},0,1}$ and $\widehat{\mathsf{C}}_{\mathsf{Ext,MMap},1,1}$ are functionally equivalent circuits with the same output distribution on all possible encodings. First of all, since the NIZK proof system is in binding mode and is perfectly simulation-extractable, any adversarially generated encoding produces the same output when extracted from, irrespective of whether $\widehat{\mathsf{C}}_{\mathsf{Ext,MMap},0,1}$ or $\widehat{\mathsf{C}}_{\mathsf{Ext,MMap},1,1}$ is used. Finally, the extraction outputs on the challenge encodings (and any encodings derived from the challenge encodings) are, by design, equal for either extraction circuit.

Thus when $b = 0$, $\mathcal{B}_7$ perfectly simulates the first sub-hybrid and when $b = 1$, $\mathcal{B}_7$ perfectly simulates the second hybrid. This completes the proof of computational indistinguishability of the first and second sub-hybrids, and hence the overall indistinguishability of hybrids 6 and 7.

Lemma 4.9. *Hybrid-8 is computationally indistinguishable from Hybrid-7 assuming that FHE is IND-CPA secure with respect to the public key-secret key pair* $(\mathsf{pk}_0, \mathsf{sk}_0)$.

Proof. Consider a PPT distinguisher $\mathcal{B}_8$ against the IND-CPA security of FHE with respect to the key pair $(\mathsf{pk}_0, \mathsf{sk}_0)$. $\mathcal{B}_8$ simulates the challenger in Hybrid-1 and interacts with the SXDH adversary $\mathcal{A}$. More specifically, $\mathcal{B}_8$ sets up the public parameters for the MMap exactly as the challenger does in Hybrid-1. In particular, it samples an additional key pair $(\mathsf{pk}_1, \mathsf{sk}_1)$ for the FHE scheme, and uses it to generate the piO obfuscated circuits for addition, inversion, multiplication and extraction over encodings. Note that it does not have not access to sk_0, but this is not a hindrance as sk_0 is not embedded in any of the aforementioned circuits in Hybrid-1.

Next, when generating the challenge encodings, it creates the following sets of plaintext-pairs for the FHE scheme:

Set-0	**Set-1**
$((\alpha_0 - \mu_0 \cdot \gamma_0), \mu_0, 0, 0, \mathbf{i})$	$(\alpha_0, 0, 0, 0, \mathbf{i})$
$((\alpha_1 - \mu_1 \cdot \gamma_1), 0, \mu_1, 0, 0, \mathbf{i})$	$(\alpha_1, 0, 0, 0, \mathbf{i})$
$((\alpha_0 - \mu_0 \cdot \gamma_0) \cdot (\alpha_1 - \mu_1 \cdot \gamma_1), (\alpha_1 \cdot \mu_0), (\alpha_0 \cdot \mu_1), -(\mu_0 \cdot \mu_1), \mathbf{i})$	$((\alpha_0 \cdot \alpha_1 + \mu_0 \cdot \mu_1 \cdot (\gamma_0 \cdot \gamma_1 - \gamma^*)), 0, 0, 0, \mathbf{i})$

where $\alpha_0, \alpha_1, \mu_0, \mu_1, \gamma_0$ and γ_1 are distributed exactly as in Hybrids 1 and 2. Next, $\mathcal{B}_8$ proceeds as follows:

1. For generating the first set of FHE ciphertexts in the challenge encodings, it forwards each pair of plaintexts as described above to the challenger in the FHE CPA-security experiment, and receives an encryption of the messages from Set-b for some unknown bit b under the key pair $(\mathsf{pk}_0, \mathsf{sk}_0)$.

2. For generating the second set of FHE ciphertexts in the challenge encodings, it encrypts the messages from Set-0 under the key pair $(\mathsf{pk}_1, \mathsf{sk}_1)$.

$\mathcal{B}_8$ then generates a *simulated* NIZK proof π for each challenge encoding using the simulation crs and simulation trapdoor $\mathsf{t}_{\mathcal{S}}$ of the NIZK proof system (guaranteed by the zero-knowledge property). Finally, it forwards to the adversary $\mathcal{A}$ the challenge encodings consisting of the FHE ciphertexts as created above and the simulated NIZK proofs.

Eventually, $\mathcal{A}$ outputs a bit b'. $\mathcal{B}_8$ outputs b' as a guess for the bit b used by the challenger in the FHE CPA-security experiment.

Note that when $b = 0$, $\mathcal{B}_8$ perfectly simulates Hybrid-7 and when $b = 1$, $\mathcal{B}_8$ perfectly simulates Hybrid-8. This completes the proof of indistinguishability of hybrids 7 and 8.

Lemma 4.10. *Hybrid-9 is computationally indistinguishable from Hybrid-8 assuming: (a) that deciding subset membership with respect to $\mathcal{L} \subset \mathcal{X}$ is computationally hard, (b) that NIZK satisfies* crs*-indistinguishability, perfect simulation-extractability under a binding* crs *and perfect zero-knowledge under a hiding* crs*, and (c) that piO satisfies probabilistic-iO security.*

The proof of this lemma follows via exactly the same set of hybrids used in the proof of Lemma 4.2, albeit in the reerse direction. The proofs of indistinguishability for each pair consecutive sub-hybrids still follow via the same set of arguments, and are hence not detailed.

This completes the proof of Theorem 4.1.

4.8 Proof of Joint-SXDH

For any $T = \mathrm{poly}(\lambda)$, let $\{\mathbf{i}_t\}_{t\in[T]}$ be a set of *arbitrary* valid level-vectors $\mathbf{i}_t$ such that no two level-vectors in the set are pairing compatible. In other words, suppose that for each $t_0, t_1 \in [T]$, $\mathbf{i}_{t_0} + \mathbf{i}_{t_1} > \mathbf{n}$. The T-joint-SXDH assumption over a degree-N asymmetric MMap requires that the following indistinguishability holds:

$$\left\{\left([\alpha_0]_{\mathbf{i}_t}, [\alpha_1]_{\mathbf{i}_t}, [\alpha_0 \cdot \alpha_1]_{\mathbf{i}_t}\right)\right\}_{t\in[T]} \overset{c}{\approx} \left\{\left([\alpha_0]_{\mathbf{i}_t}, [\alpha_1]_{\mathbf{i}_t}, [\alpha^*]_{\mathbf{i}_t}\right)\right\}_{t\in[T]},$$

where $\alpha_0, \alpha_1, \alpha^* \leftarrow \mathbb{Z}_q$.

In this subsection, we show how to extend our SXDH proof strategy to prove hardness of joint-SXDH over our proposed MMap construction via a sequence of hybrids, summarized in Table 2. Formally, we state and prove the following theorem:

Theorem 4.11. *Our MMap construction is joint-SXDH-hard assuming: (a) that deciding subset membership with respect to $\mathcal{L} \subset \mathcal{X}$ is computationally hard, (b) that NIZK satisfies* crs*-indistinguishability, perfect simulation-extractability under a binding* crs *and perfect zero-knowledge under a hiding* crs*, (c) that piO satisfies probabilistic-iO security, (d) that FHE is IND-CPA secure, and (e) that the group $\mathbb{G}$ is DDH-hard.*

Hybrid-0. In this hybrid, the public parameter pp is generated as in the real scheme and the adversary is provided with a valid joint-SXDH instance of the form:

$$\left\{\left([\alpha_0]_{\mathbf{i}_t}, [\alpha_1]_{\mathbf{i}_t}, [\alpha_0 \cdot \alpha_1]_{\mathbf{i}_t}\right)\right\}_{t\in[T]},$$

where each encoding is generated as per the real encoding algorithm.

Table 2: Overview of hybrids for proof of joint-SXDH.

Hybrid	crs	y	$\mathsf{C}_{\phi,\mathsf{MMap}}$ knows for $\phi \in \{\mathsf{Add}, \mathsf{Inv}, \mathsf{Mult}\}$	$\mathsf{C}_{\mathsf{Ext},\mathsf{MMap}}$ knows	(g, h, g', h') is of the form	First FHE Ciphertexts are encryptions of	Second FHE Ciphertexts are encryptions of	Comments
0	binding	$\notin \mathcal{L}$	$(\mathsf{sk}_0, \mathsf{sk}_1, \mathsf{t})$	$(\mathsf{sk}_0, \mathsf{sk}_1, \mathsf{t})$	$(g, g^{\gamma_0}, g^{\gamma_1}, g^{\gamma^*})$	$(\alpha_0, 0, 0, 0)$ $(\alpha_1, 0, 0, 0)$ $(\alpha_0 \cdot \alpha_1, 0, 0, 0)$	$(\alpha_0, 0, 0, 0)$ $(\alpha_1, 0, 0, 0)$ $(\alpha_0 \cdot \alpha_1, 0, 0, 0)$	—
1	hiding	$\in \mathcal{L}$	—	sk_1	$(g, g^{\gamma_0}, g^{\gamma_1}, g^{\gamma_0 \cdot \gamma_1})$	$(\alpha_0, 0, 0, 0)$ $(\alpha_1, 0, 0, 0)$ $(\alpha_0 \cdot \alpha_1, 0, 0, 0)$	$(\alpha_0, 0, 0, 0)$ $(\alpha_1, 0, 0, 0)$ $(\alpha_0 \cdot \alpha_1, 0, 0, 0)$	NIZK security + subspace-membership hardness + piO security
2-t for $t \in [T]$	hiding	$\in \mathcal{L}$	—	sk_1	$(g, g^{\gamma_0}, g^{\gamma_1}, g^{\gamma_0 \cdot \gamma_1})$	$((\alpha_0 - \mu_0 \cdot \gamma_0), \mu_0, 0, 0)$ $((\alpha_1 - \mu_1 \cdot \gamma_1), 0, \mu_1, 0, 0)$ $((\alpha_0 - \mu_0 \cdot \gamma_0) \cdot (\alpha_1 - \mu_1 \cdot \gamma_1), (\alpha_1 \cdot \mu_0), (\alpha_0 \cdot \mu_1), -(\mu_0 \cdot \mu_1))$	$(\alpha_0, 0, 0, 0)$ $(\alpha_1, 0, 0, 0)$ $(\alpha_0 \cdot \alpha_1, 0, 0, 0)$	FHE IND-CPA security
3	hiding	$\in \mathcal{L}$	—	sk_0	$(g, g^{\gamma_0}, g^{\gamma_1}, g^{\gamma_0 \cdot \gamma_1})$	$((\alpha_0 - \mu_0 \cdot \gamma_0), \mu_0, 0, 0)$ $((\alpha_1 - \mu_1 \cdot \gamma_1), 0, \mu_1, 0, 0)$ $((\alpha_0 - \mu_0 \cdot \gamma_0) \cdot (\alpha_1 - \mu_1 \cdot \gamma_1), (\alpha_1 \cdot \mu_0), (\alpha_0 \cdot \mu_1), -(\mu_0 \cdot \mu_1))$	$(\alpha_0, 0, 0, 0)$ $(\alpha_1, 0, 0, 0)$ $(\alpha_0 \cdot \alpha_1, 0, 0, 0)$	NIZK security + piO security
4-t for $t \in [T]$	hiding	$\in \mathcal{L}$	—	sk_0	$(g, g^{\gamma_0}, g^{\gamma_1}, g^{\gamma_0 \cdot \gamma_1})$	$((\alpha_0 - \mu_0 \cdot \gamma_0), \mu_0, 0, 0)$ $((\alpha_1 - \mu_1 \cdot \gamma_1), 0, \mu_1, 0, 0)$ $((\alpha_0 - \mu_0 \cdot \gamma_0) \cdot (\alpha_1 - \mu_1 \cdot \gamma_1), (\alpha_1 \cdot \mu_0), (\alpha_0 \cdot \mu_1), -(\mu_0 \cdot \mu_1))$	$((\alpha_0 - \mu_0 \cdot \gamma_0), \mu_0, 0, 0)$ $((\alpha_1 - \mu_1 \cdot \gamma_1), 0, \mu_1, 0, 0)$ $((\alpha_0 - \mu_0 \cdot \gamma_0) \cdot (\alpha_1 - \mu_1 \cdot \gamma_1), (\alpha_1 \cdot \mu_0), (\alpha_0 \cdot \mu_1), -(\mu_0 \cdot \mu_1))$	FHE IND-CPA security
5	hiding	$\in \mathcal{L}$	—	sk_0	$(g, g^{\gamma_0}, g^{\gamma_1}, g^{\gamma^*})$	$((\alpha_0 - \mu_0 \cdot \gamma_0), \mu_0, 0, 0)$ $((\alpha_1 - \mu_1 \cdot \gamma_1), 0, \mu_1, 0, 0)$ $((\alpha_0 - \mu_0 \cdot \gamma_0) \cdot (\alpha_1 - \mu_1 \cdot \gamma_1), (\alpha_1 \cdot \mu_0), (\alpha_0 \cdot \mu_1), -(\mu_0 \cdot \mu_1))$	$((\alpha_0 - \mu_0 \cdot \gamma_0), \mu_0, 0, 0)$ $((\alpha_1 - \mu_1 \cdot \gamma_1), 0, \mu_1, 0, 0)$ $((\alpha_0 - \mu_0 \cdot \gamma_0) \cdot (\alpha_1 - \mu_1 \cdot \gamma_1), (\alpha_1 \cdot \mu_0), (\alpha_0 \cdot \mu_1), -(\mu_0 \cdot \mu_1))$	DDH
6-t for $t \in [T]$	hiding	$\in \mathcal{L}$	—	sk_0	$(g, g^{\gamma_0}, g^{\gamma_1}, g^{\gamma^*})$	$((\alpha_0 - \mu_0 \cdot \gamma_0), \mu_0, 0, 0)$ $((\alpha_1 - \mu_1 \cdot \gamma_1), 0, \mu_1, 0, 0)$ $((\alpha_0 - \mu_0 \cdot \gamma_0) \cdot (\alpha_1 - \mu_1 \cdot \gamma_1), (\alpha_1 \cdot \mu_0), (\alpha_0 \cdot \mu_1), -(\mu_0 \cdot \mu_1))$	$(\alpha_0, 0, 0, 0)$ $(\alpha_1, 0, 0, 0)$ $((\alpha_0 \cdot \alpha_1 + \mu_0 \cdot \mu_1 \cdot (\gamma_0 \cdot \gamma_1 - \gamma^*)), 0, 0, 0)$	FHE IND-CPA security
7	hiding	$\in \mathcal{L}$	—	sk_1	$(g, g^{\gamma_0}, g^{\gamma_1}, g^{\gamma^*})$	$((\alpha_0 - \mu_0 \cdot \gamma_0), \mu_0, 0, 0)$ $((\alpha_1 - \mu_1 \cdot \gamma_1), 0, \mu_1, 0, 0)$ $((\alpha_0 - \mu_0 \cdot \gamma_0) \cdot (\alpha_1 - \mu_1 \cdot \gamma_1), (\alpha_1 \cdot \mu_0), (\alpha_0 \cdot \mu_1), -(\mu_0 \cdot \mu_1))$	$(\alpha_0, 0, 0, 0)$ $(\alpha_1, 0, 0, 0)$ $((\alpha_0 \cdot \alpha_1 + \mu_0 \cdot \mu_1 \cdot (\gamma_0 \cdot \gamma_1 - \gamma^*)), 0, 0, 0)$	NIZK security + piO security
8-t for $t \in [T]$	hiding	$\in \mathcal{L}$	—	sk_1	$(g, g^{\gamma_0}, g^{\gamma_1}, g^{\gamma^*})$	$(\alpha_0, 0, 0, 0)$ $(\alpha_1, 0, 0, 0)$ $((\alpha_0 \cdot \alpha_1 + \mu_0 \cdot \mu_1 \cdot (\gamma_0 \cdot \gamma_1 - \gamma^*)), 0, 0, 0)$	$(\alpha_0, 0, 0, 0)$ $(\alpha_1, 0, 0, 0)$ $((\alpha_0 \cdot \alpha_1 + \mu_0 \cdot \mu_1 \cdot (\gamma_0 \cdot \gamma_1 - \gamma^*)), 0, 0, 0)$	FHE IND-CPA security
9	binding	$\notin \mathcal{L}$	$(g, g^{\gamma_0}, g^{\gamma_1}, g^{\gamma^*})$	$(\mathsf{sk}_0, \mathsf{sk}_1, \mathsf{t})$	$(\mathsf{sk}_0, \mathsf{sk}_1, \mathsf{t})$	$(\alpha_0, 0, 0, 0)$ $(\alpha_1, 0, 0, 0)$ $((\alpha_0 \cdot \alpha_1 + \mu_0 \cdot \mu_1 \cdot (\gamma_0 \cdot \gamma_1 - \gamma^*)), 0, 0, 0)$	$(\alpha_0, 0, 0, 0)$ $(\alpha_1, 0, 0, 0)$ $((\alpha_0 \cdot \alpha_1 + \mu_0 \cdot \mu_1 \cdot (\gamma_0 \cdot \gamma_1 - \gamma^*)), 0, 0, 0)$	NIZK security + subspace-membership hardness + piO security

Hybrid-1. In this hybrid, we change the public parameter pp as follows:

1. Switch crs for the NIZK proof system to hiding mode from binding mode.
2. Switch $y \in \mathcal{X}$ from a uniform non-member instance to a uniform member instance for the language $\mathcal{L}$ with witness wit_y.
3. Compute $\mathsf{C}_{\mathsf{Add},\mathsf{piO}} = \mathsf{piO.Obf}(\widehat{\mathsf{C}}_{\mathsf{Add},\mathsf{MMap}})$, where $\widehat{\mathsf{C}}_{\mathsf{Add},\mathsf{MMap}}$ is as described in Figure 9.
4. Compute $\mathsf{C}_{\mathsf{Inv},\mathsf{piO}} = \mathsf{piO.Obf}(\widehat{\mathsf{C}}_{\mathsf{Inv},\mathsf{MMap}})$, where $\widehat{\mathsf{C}}_{\mathsf{Inv},\mathsf{MMap}}$ is as described in Figure 10.
5. Compute $\mathsf{C}_{\mathsf{Mult},\mathsf{piO}} = \mathsf{piO.Obf}(\widehat{\mathsf{C}}_{\mathsf{Mult},\mathsf{MMap}})$, where $\widehat{\mathsf{C}}_{\mathsf{Mult},\mathsf{MMap}}$ is as described in Figure 11.
6. Compute $\mathsf{C}_{\mathsf{Ext},\mathsf{piO}} = \mathsf{piO.Obf}(\widehat{\mathsf{C}}_{\mathsf{Ext},\mathsf{MMap},1,0})$, where $\widehat{\mathsf{C}}_{\mathsf{Ext},\mathsf{MMap},1,0}$ is as described in Figure 12.

Lemma 4.12. *Hybrid-1 is computationally indistinguishable from Hybrid-0 assuming: (a) that deciding subset membership with respect to $\mathcal{L} \subset \mathcal{X}$ is computationally hard, (b) that NIZK satisfies* crs*-indistinguishability, perfect simulation-extractability under a binding* crs *and perfect zero-knowledge under a hiding* crs*, and (c) that piO satisfies probabilistic-iO security.*

The proof of this lemma is identical to the proof of Lemma 4.2.

Hybrid-2-t. For each $t \in [0, T]$, Hybrid $2 - t$ changes the t^{th} set of challenge encodings (Hybrid-$2 - 0$ is identical to Hybrid-1). Suppose that the DDH instance (g, h, g', h') in the extraction circuit $\widehat{C}_{\mathsf{Ext},\mathsf{MMap},1,0}$ is distributed as:

$$(g, h, g', h') = (g, g^{\gamma_0}, g^{\gamma_1}, g^{\gamma_0 \cdot \gamma_1}).$$

Switch the plaintext underlying the *first* FHE ciphertext in the t^{th} set of challenge encodings as follows (where μ_0 and μ_1 are sampled uniformly from $\mathbb{Z}_q^*$):

$$\begin{aligned}(\alpha_0, 0, 0, 0, \mathbf{i}) &\rightarrow ((\alpha_0 - \mu_0 \cdot \gamma_0), \mu_0, 0, 0, \mathbf{i}) \\ (\alpha_1, 0, 0, 0, \mathbf{i}) &\rightarrow ((\alpha_1 - \mu_1 \cdot \gamma_1), 0, \mu_1, 0, 0, \mathbf{i}) \\ (\alpha_0 \cdot \alpha_1, 0, 0, 0, \mathbf{i}) &\rightarrow ((\alpha_0 - \mu_0 \cdot \gamma_0) \cdot (\alpha_1 - \mu_1 \cdot \gamma_1), (\alpha_1 \cdot \mu_0), (\alpha_0 \cdot \mu_1), -(\mu_0 \cdot \mu_1), \mathbf{i})\end{aligned}$$

Lemma 4.13. *For each $t \in [T]$, Hybrid-$2 - t$ is computationally indistinguishable from Hybrid-$2 - (t-1)$ assuming that FHE is IND-CPA secure with respect to the public key-secret key pair $(\mathsf{pk}_0, \mathsf{sk}_0)$.*

The proof of this lemma is identical to the proof of Lemma 4.3.

Hybrid-3. In this hybrid, we change the public parameter pp as follows: compute $\mathsf{C}_{\mathsf{Ext,piO}} = \mathsf{piO.Obf}(\widehat{\mathsf{C}}_{\mathsf{Ext,MMap},0,0})$, where $\widehat{\mathsf{C}}_{\mathsf{Ext,MMap},0,0}$ is as defined in Figure 12.

Lemma 4.14. *Hybrid-3 is computationally indistinguishable from Hybrid-2-T assuming: (a) that NIZK satisfies crs-indistinguishability, perfect simulation-extractability under a binding crs and perfect zero-knowledge under a hiding crs, and (b) that piO satisfies probabilistic-iO security.*

The proof of this lemma is identical to the proof of Lemma 4.4.

Hybrid-4-t. For each $t \in [0, T]$, Hybrid $4 - t$ changes the t^{th} set of challenge encodings (Hybrid-$4 - 0$ is identical to Hybrid-3). Again, suppose that the DDH instance (g, h, g', h') in the extraction circuit $\widehat{C}_{\mathsf{Ext,MMap},0,0}$ is distributed as:

$$(g, h, g', h') = (g, g^{\gamma_0}, g^{\gamma_1}, g^{\gamma_0 \cdot \gamma_1}).$$

Switch the plaintext underlying the *second* FHE ciphertext in the t^{th} set of challenge encodings as follows (where μ_0 and μ_1 are sampled uniformly from $\mathbb{Z}_q^*$):

$$\begin{aligned}(\alpha_0, 0, 0, 0, \mathbf{i}) &\rightarrow ((\alpha_0 - \mu_0 \cdot \gamma_0), \mu_0, 0, 0, \mathbf{i}) \\ (\alpha_1, 0, 0, 0, \mathbf{i}) &\rightarrow ((\alpha_1 - \mu_1 \cdot \gamma_1), 0, \mu_1, 0, 0, \mathbf{i}) \\ (\alpha_0 \cdot \alpha_1, 0, 0, 0, \mathbf{i}) &\rightarrow ((\alpha_0 - \mu_0 \cdot \gamma_0) \cdot (\alpha_1 - \mu_1 \cdot \gamma_1), (\alpha_1 \cdot \mu_0), (\alpha_0 \cdot \mu_1), -(\mu_0 \cdot \mu_1), \mathbf{i})\end{aligned}$$

Lemma 4.15. *For each $t \in [T]$, Hybrid-$4 - t$ is computationally indistinguishable from Hybrid-$4 - (t-1)$ assuming that FHE is IND-CPA secure with respect to the public key-secret key pair $(\mathsf{pk}_1, \mathsf{sk}_1)$.*

The proof of this lemma is identical to the proof of Lemma 4.5.

Hybrid-5. In this hybrid, we change the public parameter pp as follows: compute $\mathsf{C}_{\mathsf{Ext,piO}} = \mathsf{piO.Obf}(\widehat{\mathsf{C}}_{\mathsf{Ext,MMap},0,1})$, where $\widehat{\mathsf{C}}_{\mathsf{Ext,MMap},0,1}$ is as defined in Figure 12. Note that the only difference between the extraction circuits $\widehat{\mathsf{C}}_{\mathsf{Ext,MMap},0,0}$ and $\widehat{\mathsf{C}}_{\mathsf{Ext,MMap},0,1}$ is that the former embeds a valid DDH tuple over the group $\mathbb{G}$ while the latter embeds a uniformly random four-tuple of group elements over the group $\mathbb{G}$.

Lemma 4.16. *Hybrid-5 is computationally indistinguishable from Hybrid-4 assuming that the DDH assumption holds over the group $\mathbb{G}$.*

The proof of this lemma is identical to the proof of Lemma 4.6.

Hybrid-6-t. For each $t \in [0, T]$, Hybrid $6 - t$ changes the t^{th} set of challenge encodings (Hybrid-$6 - 0$ is identical to Hybrid-5). Recall that in hybrid-5, (g, h, g', h') in the extraction circuit $\widehat{C}_{\mathsf{Ext,MMap},0,1}$ is distributed as:

$$(g, h, g', h') = (g, g^{\gamma_0}, g^{\gamma_1}, g^{\gamma^*}).$$

Switch back the plaintext underlying the *second* FHE ciphertext in the t^{th} set of challenge encodings as follows (where μ_0 and μ_1 are sampled uniformly from $\mathbb{Z}_q^*$):

$$\begin{aligned}((\alpha_0 - \mu_0 \cdot \gamma_0), \mu_0, 0, 0, \mathbf{i}) &\rightarrow (\alpha_0, 0, 0, 0, \mathbf{i})\\ ((\alpha_1 - \mu_1 \cdot \gamma_1), 0, \mu_1, 0, 0, \mathbf{i}) &\rightarrow (\alpha_1, 0, 0, 0, \mathbf{i})\\ ((\alpha_0 - \mu_0 \cdot \gamma_0) \cdot (\alpha_1 - \mu_1 \cdot \gamma_1), (\alpha_1 \cdot \mu_0), (\alpha_0 \cdot \mu_1), -(\mu_0 \cdot \mu_1), \mathbf{i}) &\rightarrow ((\alpha_0 \cdot \alpha_1 + \mu_0 \cdot \mu_1 \cdot (\gamma_0 \cdot \gamma_1 - \gamma^*)), 0, 0, 0, \mathbf{i})\end{aligned}$$

Lemma 4.17. *For each $t \in [T]$, Hybrid-$6 - t$ is computationally indistinguishable from Hybrid-$6 - (t - 1)$ assuming that FHE is IND-CPA secure with respect to the public key-secret key pair $(\mathsf{pk}_1, \mathsf{sk}_1)$.*

The proof of this lemma is identical to the proof of Lemma 4.7.

Hybrid-7. In this hybrid, we change the public parameter pp as follows: compute $\mathsf{C}_{\mathsf{Ext,piO}} = \mathsf{piO.Obf}(\widehat{\mathsf{C}}_{\mathsf{Ext,MMap},1,1})$, where $\widehat{\mathsf{C}}_{\mathsf{Ext,MMap},1,1}$ is as defined in Figure 12.

Lemma 4.18. *Hybrid-7 is computationally indistinguishable from Hybrid-6 assuming: (a) that NIZK satisfies* crs*-indistinguishability, perfect simulation-extractability under a binding* crs *and perfect zero-knowledge under a hiding* crs*, and (b) that piO satisfies probabilistic-iO security.*

The proof of this lemma is identical to the proof of Lemma 4.8.

Hybrid-8-t. For each $t \in [0, T]$, Hybrid $8 - t$ changes the t^{th} set of challenge encodings (Hybrid-$8 - 0$ is identical to Hybrid-7). Recall that in hybrid-7, (g, h, g', h') in the extraction circuit $\widehat{C}_{\mathsf{Ext,MMap},1,1}$ is distributed as:

$$(g, h, g', h') = (g, g^{\gamma_0}, g^{\gamma_1}, g^{\gamma^*}).$$

Switch back the plaintext underlying the *first* FHE ciphertext in the t^{th} set of challenge encodings as follows (where μ_0 and μ_1 are sampled uniformly from $\mathbb{Z}_q^*$):

$$\begin{aligned}((\alpha_0 - \mu_0 \cdot \gamma_0), \mu_0, 0, 0, \mathbf{i}) &\rightarrow (\alpha_0, 0, 0, 0, \mathbf{i})\\ ((\alpha_1 - \mu_1 \cdot \gamma_1), 0, \mu_1, 0, 0, \mathbf{i}) &\rightarrow (\alpha_1, 0, 0, 0, \mathbf{i})\\ ((\alpha_0 - \mu_0 \cdot \gamma_0) \cdot (\alpha_1 - \mu_1 \cdot \gamma_1), (\alpha_1 \cdot \mu_0), (\alpha_0 \cdot \mu_1), -(\mu_0 \cdot \mu_1), \mathbf{i}) &\rightarrow ((\alpha_0 \cdot \alpha_1 + \mu_0 \cdot \mu_1 \cdot (\gamma_0 \cdot \gamma_1 - \gamma^*)), 0, 0, 0, \mathbf{i})\end{aligned}$$

Lemma 4.19. *For each $t \in [T]$, Hybrid-$8 - t$ is computationally indistinguishable from Hybrid-$8 - (t - 1)$ assuming that FHE is IND-CPA secure with respect to the public key-secret key pair $(\mathsf{pk}_0, \mathsf{sk}_0)$.*

The proof of this lemma is identical to the proof of Lemma 4.9.

Hybrid-9. In this hybrid, we change the public parameter pp as follows:

1. Switch crs for the NIZK proof system to binding mode, as in the real scheme
2. Switch $y \in \mathcal{X}$ to a uniform non-member instance, as in the real scheme.
3. Compute $\mathsf{C}_{\mathsf{Add,piO}} = \mathsf{piO.Obf}(\mathsf{C}_{\mathsf{Add,MMap}})$, as in the real scheme.
4. Compute $\mathsf{C}_{\mathsf{Inv,piO}} = \mathsf{piO.Obf}(\mathsf{C}_{\mathsf{Inv,MMap}})$, as in the real scheme.

5. Compute $\mathsf{C}_{\mathsf{Mult},\mathsf{piO}} = \mathsf{piO.Obf}(\mathsf{C}_{\mathsf{Mult},\mathsf{MMap}})$, as in the real scheme.

6. Compute $\mathsf{C}_{\mathsf{Ext},\mathsf{piO}} = \mathsf{piO.Obf}(\mathsf{C}_{\mathsf{Ext},\mathsf{MMap}})$, as in the real scheme.

Lemma 4.20. *Hybrid-9 is computationally indistinguishable from Hybrid-8 assuming: (a) that deciding subset membership with respect to $\mathcal{L} \subset \mathcal{X}$ is computationally hard, (b) that NIZK satisfies* crs*-indistinguishability, perfect simulation-extractability under a binding* crs *and perfect zero-knowledge under a hiding* crs*, and (c) that piO satisfies probabilistic-iO security.*

The proof of this lemma is identical to the proof of Lemma 4.10.

Finally, Hybrid-9 is identical to the game where the adversary is given a random invalid SXDH instance generated as per the real scheme. This completes the proof of Theorem 4.11.

5 Constructing a Squared-DDH-Hard Asymmetric MMap

It is easy to extend the approach used for our MMap construction in Section 4 to construct a squared-DDH (SDDH)-hard asymmetric MMap. Recall that the SDDH assumption over a degree-N asymmetric MMap requires that the following indistinguishability holds for any level-vector $\mathbf{i}$ such that $\mathbf{0} < \mathbf{i} \leq \mathbf{n}$:

$$\left([\alpha]_{\mathbf{i}}, [\alpha^2]_{\mathbf{i}}\right) \quad \stackrel{c}{\approx} \quad \left([\alpha]_{\mathbf{i}}, [\alpha^*]_{\mathbf{i}}\right),$$

where $\alpha, \alpha^* \leftarrow \mathbb{Z}_q$. The SDDH assumption is essentially identical to the 2-exponent-DDH (EDDH) assumption. It implies the SXDH assumption (the converse is not known to be true).

In this section, we show how to construct an SDDH-hard asymmetric MMap using essentially the same set of cryptoprimitives as used by our SXDH-hard MMap construction, except for the group $\mathbb{G}$, which we now require to be SDDH-hard as opposed to just DDH-hard. More concretely, the construction relies on the following cryptoprimitives:

- A probabilistic-iO scheme $\mathsf{piO} = (\mathsf{piO.Obf}, \mathsf{piOEval})$.
- A fully-homomorphic encryption scheme $\mathsf{FHE} = (\mathsf{FHE.Gen}, \mathsf{FHE.Enc}, \mathsf{FHE.Dec}, \mathsf{FHE.Eval})$ such that the message space is $\mathbb{Z}_q$ for some prime $q = \mathrm{poly}(\lambda)$ (λ being the security parameter).
- A simulation-extractable non-interactive $\mathsf{NIZK} = (\mathsf{NIZK.Setup}, \mathsf{NIZK.Prove}, \mathsf{NIZK.Verify})$.
- A pair of sets $(\mathcal{X}, \mathcal{L})$ such that $\mathcal{L} \subset \mathcal{X}$ and:
  1. Given $x \in \mathcal{X}$ it is *computationally hard* to decide if $x \in \mathcal{L}$.
  2. For each $y \in \mathcal{L}$, there exists a *unique* witness wit_y for the statement $y \in \mathcal{L}$.

  We define a relation $\mathsf{R}_{\mathcal{L}}$ as follows: $(y, \mathsf{wit}_y) \in R_{\mathcal{L}}$ if and only if wit_y is a witness for the statement $y \in \mathcal{L}$.
- A pairing-free SDDH-hard group $\mathbb{G}$ of prime order q, where $\mathcal{D}_{\mathrm{SDDH},\mathbb{G}}$ denotes the distribution of all valid SDDH samples over the group $\mathbb{G}$.

At a high level, the key change from the SXDH-hard MMap construction is that each FHE ciphertext ct_b in a given encoding encrypts a three-tuple $(a_{b,0}, a_{b,1}, a_{b,2})$ instead of a four-tuple, and the extraction circuit now embeds a three-tuple of group elements (g, h, h') instead of a four-tuple of group elements.

The crux of the proof lies in switching the plaintexts underlying the challenge encodings in the SDDH game to a form where the challenge encoding is a valid SDDH instance over the MMap when the extraction circuit embeds a valid SDDH tuple over the group $\mathbb{G}$, while the challenge encoding is a random invalid SDDH instance over the MMap when the extraction circuit embeds a random invalid SDDH tuple over the group $\mathbb{G}$. We then show via a sequence of hybrids that these two cases are indistinguishable based on a sequence of hybrids that exploit the computational security properties of the underlying cryptoprimitives and the hardness of SDDH over the group $\mathbb{G}$.

For the sake of completeness, we present the detailed construction below. In particular, we focus on the structure of the encodings, and the circuits for addition, inversion, multiplication and extraction over encodings. Note that our construction can be easily generalized to achieve ℓ-EDDH-hard asymmetric MMaps for any $\ell \geq 3$.

5.1 Encodings

Level-Zero Encodings. As in the previous construction, level-$\mathbf{0}$ encodings are treated slightly different from encodings at other "non-zero" levels. In particular, our level-$\mathbf{0}$ encodings are equipped with their own set of algorithms for encoding, manipulation, extraction and zero-testing. We informally mention these for the sake of completeness.

The level-$\mathbf{0}$ encoding of a plaintext element $a \in \mathbb{Z}_q$ is a itself. Adding/multiplying two level-$\mathbf{0}$ encodings (equivalently, additively inverting a level-$\mathbf{0}$ encoding) is simply done via addition/multiplication (equivalently, additive inversion) in $\mathbb{Z}_q$. Multiplying a level-$\mathbf{0}$ encoding with any other encoding at some level $\mathbf{i}$ should result in an encoding at level-$\mathbf{i}$. This is implemented with a shift-and-add algorithm built on top of the standard encoding addition algorithm described subsequently in Section 5.2. Finally, extracting a level-$\mathbf{0}$ encoding $a \in \mathbb{Z}_q$ outputs g^a, where g is a fixed generator for the group $\mathbb{G}$. As will be clear later, this is consistent with the extraction algorithm for any other level $\mathbf{i}$. Zero-testing follows trivially.

We omit formal descriptions of operations for level-$\mathbf{0}$ encodings to ease notation and focus on the more interesting cases at "non-zero" encoding levels.

Level-i Encodings. We now describe the procedure of encoding a plaintext element at any level $\mathbf{i}$ such that $\mathbf{0} < \mathbf{i} \leq \mathbf{n}$. An encoding of a plaintext element $a \in \mathbb{Z}_q$ at level-vector $\mathbf{i}$ is a tuple of the form: $(\mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, \pi)$, where:

- ct_0 and ct_1 are FHE encryptions of the tuple $(a, 0, 0, \mathbf{i})$ under the public key-secret key pairs $(\mathsf{pk}_0, \mathsf{sk}_0)$ and $(\mathsf{pk}_1, \mathsf{sk}_1)$ respectively.
- π is a NIZK proof for the statement $\mathsf{st} = (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, y \in \mathcal{X})$ under the NP language L described in Figure 13, where $y \in \mathcal{X}$ is available as part of the public parameter for our MMap scheme.

Note that the additional zero entries encrypted in each encoding appear redundant, but will be useful later in the proof of security.

Language L:

Statement: The statement st is as follows: $\mathsf{st} = (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, y \in \mathcal{X})$

Witness: The witness has one of the following three forms:

1. $\mathsf{wit} = (\mathsf{sk}_0, \mathsf{sk}_1)$
2. $\mathsf{wit} = (\mathsf{m}, \mathsf{r}_0, \mathsf{r}_1)$
3. $\mathsf{wit} = \mathsf{wit}_y$

Relation: $\mathsf{R}(\mathsf{st}, \mathsf{wit}) = 1$ if and only if:

- **Either** $\mathsf{wit} = (\mathsf{sk}_0, \mathsf{sk}_1)$ and we have:
  1. $(\mathsf{pk}_0, \mathsf{sk}_0), (\mathsf{pk}_1, \mathsf{sk}_1) \in \mathcal{K}_{\mathrm{FHE}}$.
  2. $\mathsf{FHE.Dec}(\mathsf{sk}_0, \mathsf{ct}_0) = \mathsf{FHE.Dec}(\mathsf{sk}_1, \mathsf{ct}_1) = (\mathsf{m}, 0, 0, \mathbf{i})$ for some $\mathsf{m} \in \mathbb{Z}_q$.
- **Or** $\mathsf{wit} = (\mathsf{m}, \mathsf{r}_0, \mathsf{r}_1)$ and we have:
  1. $\mathsf{m} \in \mathbb{Z}_q$.
  2. For each $b \in \{0, 1\}$, $\mathsf{FHE.Enc}(\mathsf{pk}_b, (\mathsf{m}, 0, 0, \mathbf{i}); \mathsf{r}_b) = \mathsf{ct}_b$.
- **Or** $\mathsf{wit} = \mathsf{wit}_y$ and we have: $(y, \mathsf{wit}_y) \in \mathsf{R}_{\mathcal{L}}$.

Figure 13: Let $\mathcal{K}_{\mathrm{FHE}}$ be the set of all valid key pairs under the FHE scheme. We define an NP language L as in the figure above. For simplicity, L parameterized by N – the number of levels for our MMap scheme.

5.2 Addition of Encodings

We now describe the procedure for adding two encodings. Suppose we have two encodings at the same level $\mathbf{i}$ of the form:

$$(\mathsf{ct}_{1,0}, \mathsf{ct}_{1,1}, \mathbf{i}, \pi_1), (\mathsf{ct}_{2,0}, \mathsf{ct}_{2,1}, \mathbf{i}, \pi_2).$$

Conceptually, we add these two encodings by exploiting the fully-homomorphic nature of the encryption scheme. More concretely, we homomorphically evaluate the circuit $\mathsf{C}_{\mathsf{Add,FHE}}$ (described in Figure 14) on the corresponding ciphertext components of the two input encodings to generate the ciphertext components for the output encoding. We also generate a proof for the validity of the resulting encoding using the tuple of secret keys $(\mathsf{sk}_0, \mathsf{sk}_1)$ as witness.

Inputs: A tuple of the form $((\{a_{1,\ell}\}_{\ell\in[0,2]}, \mathbf{i}_1), ((\{a_{2,\ell}\}_{\ell\in[0,2]}, \mathbf{i}_2))$, where for each $\ell \in [0, 2]$, we have $a_{1,\ell}, a_{2,\ell} \in \mathbb{Z}_q$.

Computation: Proceed as follows:

1. If $\mathbf{i}_1 \neq \mathbf{i}_2$ or $\mathbf{i}_1 \leq \mathbf{0}$ or $\mathbf{i}_1 > \mathbf{n}$, output $\perp$.
2. Else, output $(\{a_{1,\ell} + a_{2,\ell}\}_{\ell\in[0,2]}, \mathbf{i}_1)$.

Figure 14: Circuit $\mathsf{C}_{\mathsf{Add,FHE}}$

For technical reasons that are relevant to the proof of security, we check: (a) the validity of the proofs π_1 and π_2 for the language L that are provided as part of the input encodings, and (b) the "consistency" of the FHE ciphertexts that are provided as part of the input encodings, i.e., whether the ciphertexts encrypt the same underlying plaintext, and whether the plaintext is formatted as per the specifications of the scheme.

Figure 15 details the operation of the encoding-addition circuit $\mathsf{C}_{\mathsf{Add,MMap}}$.

As in the previous construction, the addition embeds multiple secrets, including the secret keys sk_0 and sk_1, as well as the extraction trapdoor t for the NIZK. Hence, the circuit is not made public as is; we only make available an obfuscated version of the circuit obtained by running the evaluation algorithm of the probabilistic iO scheme piO on it.

We emphasize that when the element $y \in \mathcal{X}$ is a non-member for the language $\mathcal{L}$, then under a binding crs, the "**If**" condition in step 4 of $\mathsf{C}_{\mathsf{Add,MMap}}$ is never satisfied. This follows from the perfect extractability guarantee of the NIZK proof system. However, the condition may be satisfied during some hybrid in the proof of security, under a hiding crs when the element $y \in \mathcal{X}$ is a member for the language $\mathcal{L}$.

5.3 Inversion of Encodings

We now describe the procedure for additively inverting an encoding. Suppose we have an encoding at the level $\mathbf{i}$ of the form: $(\mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, \pi)$. Again, conceptually, we additively invert this encoding by exploiting the fully-homomorphic nature of the encryption scheme. More concretely, we homomorphically evaluate the circuit $\mathsf{C}_{\mathsf{Inv,FHE}}$ (described in Figure 16) on the corresponding ciphertext components of the input encoding to generate the ciphertext components for the output encoding. We also generate a proof for the validity of the resulting encoding using the tuple of secret keys $(\mathsf{sk}_0, \mathsf{sk}_1)$ as witness.

Similar to the addition procedure, for technical reasons that are relevant to the proof of security, we check the validity and consistency of the input encodings. Figure 17 details the operation of the encoding-inversion circuit $\mathsf{C}_{\mathsf{Inv,MMap}}$. Agsin, since it embeds the same set of secrets as the addition circuit, the circuit $\mathsf{C}_{\mathsf{Inv,MMap}}$ is not made public as is; we only make available an obfuscated version of the circuit obtained by running the evaluation algorithm of the probabilistic iO scheme piO on it.

We again emphasize that when the element $y \in \mathcal{X}$ is a non-member for the language $\mathcal{L}$, then under a binding crs, the "**If**" condition in step 4 of $\mathsf{C}_{\mathsf{Add,MMap}}$ is never satisfied. This follows from the perfect extractability guarantee of the NIZK proof system. However, the condition may be satisfied during some hybrid in the proof of security, under a hiding crs when the element $y \in \mathcal{X}$ is a member for the language $\mathcal{L}$.

Encoding Addition Circuit: $\mathsf{C}_{\mathsf{Add,MMap}}$

Inputs: A tuple of the form: $((\mathsf{ct}_{1,0}, \mathsf{ct}_{1,1}, \mathbf{i}, \pi_1), (\mathsf{ct}_{2,0}, \mathsf{ct}_{2,1}, \mathbf{i}, \pi_2), y \in \mathcal{X})$, where the proofs π_1 and π_2 are over the language L.

Outputs: Either a tuple of the form $(\mathsf{ct}^*_0, \mathsf{ct}^*_1, \mathbf{i}, \pi^*)$, or $\perp$.

Computation: Proceed as follows:

1. **If** any of the following is true:
   (a) $\mathsf{NIZK.Verify}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_{1,0}, \mathsf{ct}_{1,1}, \mathbf{i}, y \in \mathcal{X}), \pi_1) = 0$.
   (b) $\mathsf{NIZK.Verify}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_{2,0}, \mathsf{ct}_{2,1}, \mathbf{i}, y \in \mathcal{X}), \pi_2) = 0$.

   then output $\perp$.
2. Recover the following:
$$(\{a_{1,0,\ell}\}_{\ell\in[0,2]}, \mathbf{i}_{1,0}) = \mathsf{FHE.Dec}(\mathsf{sk}_0, \mathsf{ct}_{1,0}) \quad , \quad (\{a_{1,1,\ell}\}_{\ell\in[0,2]}, \mathbf{i}_{1,1}) = \mathsf{FHE.Dec}(\mathsf{sk}_1, \mathsf{ct}_{1,1}),$$
$$(\{a_{2,0,\ell}\}_{\ell\in[0,2]}, \mathbf{i}_{2,0}) = \mathsf{FHE.Dec}(\mathsf{sk}_0, \mathsf{ct}_{2,0}) \quad , \quad (\{a_{2,1,\ell}\}_{\ell\in[0,2]}, \mathbf{i}_{2,1}) = \mathsf{FHE.Dec}(\mathsf{sk}_1, \mathsf{ct}_{2,1}).$$
3. Compute the following:
$$\mathsf{ct}^*_0 = \mathsf{FHE.Eval}(\mathsf{pk}_0, \mathsf{ct}_{1,0}, \mathsf{ct}_{2,0}, \mathsf{C}_{\mathsf{Add,FHE}}) \quad , \quad \mathsf{ct}^*_1 = \mathsf{FHE.Eval}(\mathsf{pk}_1, \mathsf{ct}_{1,1}, \mathsf{ct}_{2,1}, \mathsf{C}_{\mathsf{Add,FHE}}).$$
4. **If** any of the following is true:
   (a) $a_{1,0,0} \neq a_{1,1,0}$ or $a_{2,0,0} \neq a_{2,1,0}$.
   (b) There exists some $(t, b, \ell) \in \{1, 2\} \times \{0, 1\} \times \{1, 2\}$ such that $a_{t,b,\ell} \neq 0$.
   (c) There exists some $(t, b) \in \{1, 2\} \times \{0, 1\}$ such that $\mathbf{i}_{t,b} \neq \mathbf{i}$.

   then proceed as follows:
   (a) Extract $\mathsf{wit}_y = \mathsf{NIZK.Ext}(\mathsf{t}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_{1,0}, \mathsf{ct}_{1,1}, \mathbf{i}, y), \pi_1)$.
   (b) **If** $\mathsf{R}_{\mathcal{L}}(\mathsf{wit}_y, y) = 0$, output $\perp$.
   (c) **Else**, generate $\pi^* \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}^*_0, \mathsf{ct}^*_1, \mathbf{i}, y), \mathsf{wit}_y)$.
5. **Else**, generate $\pi^* \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}^*_0, \mathsf{ct}^*_1, \mathbf{i}, y), (\mathsf{sk}_0, \mathsf{sk}_1))$.
6. Output $(\mathsf{ct}^*_0, \mathsf{ct}^*_1, \mathbf{i}, \pi^*)$.

Figure 15: Let $\mathsf{crs} \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$. We now assume that crs is in binding mode. Let $\mathsf{NIZK.Ext}$ be the extraction algorithm for NIZK. Let t be the extraction trapdoor corresponding to crs. Additionally, let $(\mathsf{pk}_0, \mathsf{sk}_0), (\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathsf{FHE.Gen}(1^\lambda)$. We define a probabilistic circuit $\mathsf{C}_{\mathsf{Add,MMap}}$ (which has t and the key pairs $(\mathsf{pk}_0, \mathsf{sk}_0)$ and $(\mathsf{pk}_1, \mathsf{sk}_1)$ hard-wired into it) as above.

Inputs: A tuple of the form $((a_0, a_1, a_2, \mathbf{i}))$ where for each $\ell \in [0, 2]$, we have $a_\ell \in \mathbb{Z}_q$.

Computation: If $\mathbf{i} \leq \mathbf{0}$ or $\mathbf{i} > N$, output $\perp$ else output $(-a_0, -a_1, -a_2, \mathbf{i})$.

Figure 16: Circuit $\mathsf{C}_{\mathsf{Inv,FHE}}$

5.4 Multiplication of Encodings

We now describe the procedure for multiplying two encodings at levels $\mathbf{i}_1$ and $\mathbf{i}_2$, respectively such that $\mathbf{i}_1 + \mathbf{i}_2 \leq \mathbf{n}$. Suppose we have two encodings of the form:

$$(\mathsf{ct}_{1,0}, \mathsf{ct}_{1,1}, \mathbf{i}_1, \pi_1), (\mathsf{ct}_{2,0}, \mathsf{ct}_{2,1}, \mathbf{i}_2, \pi_2).$$

Conceptually, we multiply these two encodings by again exploiting the fully-homomorphic nature of the encryption scheme. More concretely, we homomorphically evaluate the circuit $\mathsf{C}_{\mathsf{Mult,FHE}}$ (described in Figure 18) on the corresponding ciphertext components of the two input encodings to generate the ciphertext components for the output encoding. We also generate a proof for the validity of the resulting encoding using the tuple of secret keys $(\mathsf{sk}_0, \mathsf{sk}_1)$ as witness.

Similar to the addition and inversion procedures, for technical reasons that are relevant to the proof of security, we

Encoding Inversion Circuit: $\mathsf{C}_{\mathsf{Inv,MMap}}$

Inputs: A tuple of the form: $(\{\mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, \pi, y \in \mathcal{X})$, where the proof π is over the language L.

Outputs: Either a tuple of the form $(\mathsf{ct}_0^*, \mathsf{ct}_1^*, \mathbf{i}, \pi^*)$, or $\perp$.

Computation: Proceed as follows:

1. If $\mathsf{NIZK.Verify}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, y), \pi) = 0$, then output $\perp$.
2. Recover the following:
$$(\{a_{0,\ell}\}_{\ell\in[0,2]}, \mathbf{i}_0) = \mathsf{FHE.Dec}(\mathsf{sk}_0, \mathsf{ct}_0) \quad , \quad (\{a_{1,\ell}\}_{\ell\in[0,2]}, \mathbf{i}_1) = \mathsf{FHE.Dec}(\mathsf{sk}_1, \mathsf{ct}_1).$$
3. Compute the following:
$$\mathsf{ct}_0^* = \mathsf{FHE.Eval}(\mathsf{pk}_0, \mathsf{ct}_0, \mathsf{C}_{\mathsf{Inv,FHE}}) \quad , \quad \mathsf{ct}_1^* = \mathsf{FHE.Eval}(\mathsf{pk}_1, \mathsf{ct}_1, \mathsf{C}_{\mathsf{Inv,FHE}}).$$
4. If any of the following is true:
   (a) $a_{0,0} \neq a_{0,1}$.
   (b) There exists some $(b, \ell) \in \{0, 1\} \times \{1, 2\}$ such that $a_{b,\ell} \neq 0$.
   (c) There exists some $b \in \{0, 1\}$ such that $\mathbf{i}_b \neq \mathbf{i}$.

   then proceed as follows:
   (a) Extract $\mathsf{wit}_y = \mathsf{NIZK.Ext}(\mathsf{t}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, y), \pi)$.
   (b) **If** $\mathsf{R}_{\mathcal{L}}(\mathsf{wit}_y, y) = 0$, output $\perp$.
   (c) **Else**, generate $\pi^* \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0^*, \mathsf{ct}_1^*, \mathbf{i}, y), \mathsf{wit}_y)$.
5. **Else**, generate $\pi^* \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0^*, \mathsf{ct}_1^*, \mathbf{i}, y), (\mathsf{sk}_0, \mathsf{sk}_1))$.
6. Output $(\mathsf{ct}_0^*, \mathsf{ct}_1^*, \mathbf{i}, \pi^*)$.

Figure 17: Let $\mathsf{crs} \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$. We now assume that crs is in binding mode. Let $\mathsf{NIZK.Ext}$ be the extraction algorithm for NIZK. Let t be the extraction trapdoor corresponding to crs. Additionally, let $(\mathsf{pk}_0, \mathsf{sk}_0), (\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathsf{FHE.Gen}(1^\lambda)$. We define a probabilistic circuit $\mathsf{C}_{\mathsf{Inv,MMap}}$ (which has t and the key pairs $(\mathsf{pk}_0, \mathsf{sk}_0)$ and $(\mathsf{pk}_1, \mathsf{sk}_1)$ hard-wired into it) as above.

Inputs: A tuple of the form $((\{a_{1,\ell}\}_{\ell\in[0,2]}, \mathbf{i}_1), ((\{a_{2,\ell}\}_{\ell\in[0,2]}, \mathbf{i}_2))$, where for each $\ell \in [0, 2]$, we have $a_{1,\ell}, a_{2,\ell} \in \mathbb{Z}_q$.

Computation: Proceed as follows:

- If $\mathbf{i}_1 \leq \mathbf{0}$ or $\mathbf{i}_2 \leq \mathbf{0}$ or $\mathbf{i}_1 + \mathbf{i}_2 > \mathbf{n}$, output $\perp$.
- Else if $(a_{1,1}, a_{1,2}) = (0, 0)$ then output $((\{a_{1,0} \cdot a_{2,\ell}\}_{\ell\in[0,2]}, i_1, j_2))$.
- Else if $(a_{2,1}, a_{2,2}) = (0, 0)$ then output $((\{a_{2,0} \cdot a_{1,\ell}\}_{\ell\in[0,2]}, i_1, j_2))$.
- Else output $\perp$.

Figure 18: Circuit $\mathsf{C}_{\mathsf{Mult,FHE}}$

check the validity and consistency of the input encodings. Figure 19 details the operation of the encoding-multiplication circuit $\mathsf{C}_{\mathsf{Mult,MMap}}$.

As in the previous construction, the multiplication circuit embeds multiple secrets, including the secret keys sk_0 and sk_1, as well as the extraction trapdoor t for the NIZK. Hence, the circuit is not made public as is; we only make available an obfuscated version of the circuit obtained by running the evaluation algorithm of the probabilistic iO scheme piO on it.

We emphasize that when the element $y \in \mathcal{X}$ is a non-member for the language $\mathcal{L}$, then under a binding crs, the "**If**" condition in step 4 of $\mathsf{C}_{\mathsf{Mult,MMap}}$ is never satisfied. This follows from the perfect extractability guarantee of the NIZK proof system. However, the condition may be satisfied during some hybrid in the proof of security, under a hiding crs when the element $y \in \mathcal{X}$ is a member for the language $\mathcal{L}$.

Encoding Multiplication Circuit: $\mathsf{C}_{\mathsf{Mult},\mathbf{MMap}}$

Inputs: A tuple of the form: $((\mathsf{ct}_{1,0}, \mathsf{ct}_{1,1}, \mathbf{i}_1, \pi_1), (\mathsf{ct}_{2,0}, \mathsf{ct}_{2,1}, \mathbf{i}_2, \pi_2), y \in \mathcal{X})$, where the proofs π_1 and π_2 are over the language L.

Outputs: Either a tuple of the form $(\mathsf{ct}_0^*, \mathsf{ct}_1^*, \mathbf{i}_1 + \mathbf{i}_2, \pi^*)$, or $\perp$.

Computation: Proceed as follows:

1. **If** any of the following is true:
   (a) $\mathsf{NIZK.Verify}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_{1,0}, \mathsf{ct}_{1,1}, \mathbf{i}_1, y \in \mathcal{X}), \pi_1) = 0$.
   (b) $\mathsf{NIZK.Verify}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_{2,0}, \mathsf{ct}_{2,1}, \mathbf{i}_2, y \in \mathcal{X}), \pi_2) = 0$.

   then output $\perp$.
2. Recover the following:
$$(\{a_{1,0,\ell}\}_{\ell\in[0,2]}, \mathbf{i}_{1,0}) = \mathsf{FHE.Dec}(\mathsf{sk}_0, \mathsf{ct}_{1,0}) \quad , \quad (\{a_{1,1,\ell}\}_{\ell\in[0,2]}, \mathbf{i}_{1,1}) = \mathsf{FHE.Dec}(\mathsf{sk}_1, \mathsf{ct}_{1,1}),$$
$$(\{a_{2,0,\ell}\}_{\ell\in[0,2]}, \mathbf{i}_{2,0}) = \mathsf{FHE.Dec}(\mathsf{sk}_0, \mathsf{ct}_{2,0}) \quad , \quad (\{a_{2,1,\ell}\}_{\ell\in[0,2]}, \mathbf{i}_{2,1}) = \mathsf{FHE.Dec}(\mathsf{sk}_1, \mathsf{ct}_{2,1}).$$
3. Compute the following:
$$\mathsf{ct}_0^* = \mathsf{FHE.Eval}(\mathsf{pk}_0, \mathsf{ct}_{1,0}, \mathsf{ct}_{2,0}, \mathsf{C}_{\mathsf{Mult,FHE}}) \quad , \quad \mathsf{ct}_1^* = \mathsf{FHE.Eval}(\mathsf{pk}_1, \mathsf{ct}_{1,1}, \mathsf{ct}_{2,1}, \mathsf{C}_{\mathsf{Mult,FHE}}).$$
4. **If** any of the following is true:
   (a) $a_{1,0,0} \neq a_{1,1,0}$ or $a_{2,0,0} \neq a_{2,1,0}$.
   (b) There exists some $(t, b, \ell) \in \{1, 2\} \times \{0, 1\} \times \{1, 2\}$ such that $a_{t,b,\ell} \neq 0$.
   (c) There exists some $b \in \{0, 1\}$ such that $\mathbf{i}_{1,b} \neq \mathbf{i}_1$ or $\mathbf{i}_{2,b} \neq \mathbf{i}_2$.

   then proceed as follows:
   (a) Extract $\mathsf{wit}_y = \mathsf{NIZK.Ext}(\mathsf{t}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_{1,0}, \mathsf{ct}_{1,1}, \mathbf{i}, y), \pi_1)$.
   (b) **If** $\mathsf{R}_{\mathcal{L}}(\mathsf{wit}_y, y) = 0$, output $\perp$.
   (c) **Else**, generate $\pi^* \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0^*, \mathsf{ct}_1^*, \mathbf{i}_1 + \mathbf{i}_2, y), \mathsf{wit}_y)$.
5. **Else**, generate $\pi^* \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0^*, \mathsf{ct}_1^*, \mathbf{i}_1 + \mathbf{i}_2, y), (\mathsf{sk}_0, \mathsf{sk}_1))$.
6. Output $(\mathsf{ct}_0^*, \mathsf{ct}_1^*, \mathbf{i}_1 + \mathbf{i}_2, \pi^*)$.

Figure 19: Let $\mathsf{crs} \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$. We now assume that crs is in binding mode. Let $\mathsf{NIZK.Ext}$ be the extraction algorithm for NIZK. Let t be the extraction trapdoor corresponding to crs. Additionally, let $(\mathsf{pk}_0, \mathsf{sk}_0), (\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathsf{FHE.Gen}(1^\lambda)$. We define a probabilistic circuit $\mathsf{C}_{\mathsf{Mult,MMap}}$ (which has t and the key pairs $(\mathsf{pk}_0, \mathsf{sk}_0)$ and $(\mathsf{pk}_1, \mathsf{sk}_1)$ hard-wired into it) as in above.

5.5 Extraction and Zero-Testing

Extraction. We now describe the procedure for extracting a canonical string from an encoding at any given level-vector. Suppose we have an encodings at the level $\mathbf{i}$ of the form: $(\{\mathsf{ct}_{\ell,b}\}_{\ell\in\{0,1,2\},b\in\{0,1\}}, \mathbf{i}, \pi)$. The straightforward way of doing this is to decrypt the corresponding ciphertexts using sk_0 and sk_1, and outputting $g^* = g^{a_{0,0}} = g^{a_{1,0}}$, where g is a fixed group element embedded inside the extraction circuit. However, for technical reasons relevant to the proof of security, we sightly depart from this straightforward process.

More concretely, our extraction circuit embeds a three-tuple of group elements (g, h, h'), uses sk_0 to recover $a_{0,\ell}$ for each $\ell \in \{0, 1, 2\}$, and computes
$$g^* = g^{a_{0,0}} \cdot h^{a_{0,1}} \cdot (h')^{a_{0,2}}.$$
It is easy to see that when the encodings are honestly generated, i.e., when the "auxiliary" terms in the plaintext are all 0, this is functionally equivalent to the simpler extraction strategy described above. However, if one or more of the "auxiliary" plaintext terms is non-zero, the two processes are no longer equivalent. As it turns out, we exploit this fact in the proof of security.

Again, for technical reasons that are relevant to the proof of security, we also use the secret keys sk_0 and sk_1, as well as the NIZK extraction trapdoor t, to check: (a) the validity of the proofs π_1 and π_2 for the language L that are provided as part of the input encodings, and (b) the "consistency" of the FHE ciphertexts that are provided as part of the input encodings, i.e., whether the "auxiliary" set of plaintext elements are all zero and whether both ciphertexts encrypt

Extraction Circuit: $\mathsf{C}_{\mathsf{Ext,MMap}}$

Inputs: A tuple of the form: $((\mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, \pi), y \in \mathcal{X})$, where the proof π is over the language L.

Outputs: Either a group element g^* or $\perp$.

Computation: Proceed as follows:

1. If $\mathsf{NIZK.Verify}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, y), \pi) = 0$, then output $\perp$.
2. Recover the following:
$$(\{a_{0,\ell}\}_{\ell \in [0,2]}, \mathbf{i}_0) = \mathsf{FHE.Dec}(\mathsf{sk}_0, \mathsf{ct}_0) \quad , \quad (\{a_{1,\ell}\}_{\ell \in [0,2]}, \mathbf{i}_1) = \mathsf{FHE.Dec}(\mathsf{sk}_1, \mathsf{ct}_1).$$
3. Compute $g^* = g^{a_{0,0}} \cdot h^{a_{0,1}} \cdot (h')^{a_{0,2}}$.
4. If any of the following is true:
   (a) $a_{0,0} \neq a_{0,1}$.
   (b) There exists some $(b, \ell) \in \{0,1\} \times \{1,2\}$ such that $a_{b,\ell} \neq 0$.
   (c) There exists some $b \in \{0,1\}$ such that $\mathbf{i}_b \neq \mathbf{i}$.

   then proceed as follows:
   (a) Extract $\mathsf{wit}_y = \mathsf{NIZK.Ext}(\mathsf{t}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, y), \pi)$.
   (b) **If** $\mathsf{R}_{\mathcal{L}}(\mathsf{wit}_y, y) = 0$, output $\perp$.
5. **Else**, output g^*.

Figure 20: Let $\mathsf{crs} \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$. We now assume that crs is in binding mode. Let $\mathsf{NIZK.Ext}$ be the extraction algorithm for NIZK. Let t be the extraction trapdoor corresponding to crs. Additionally, let $(\mathsf{pk}_0, \mathsf{sk}_0), (\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathsf{FHE.Gen}(1^\lambda)$, and let $g, h, h' \in \mathbb{G}$ be arbitrary group elements. We define a deterministic circuit $\mathsf{C}_{\mathsf{Ext,MMap}}$ (which has t, the key pairs $(\mathsf{pk}_0, \mathsf{sk}_0)$ and $(\mathsf{pk}_1, \mathsf{sk}_1)$ and the collection of group elements (g, h, h') hard-wired into it) as above.

the same level.

Figure 20 details the extraction circuit $\mathsf{C}_{\mathsf{Ext,MMap}}$. The circuit is not made public as is; we only make available an obfuscated version of the circuit obtained by running the evaluation algorithm of the probabilistic iO scheme piO on it.

Finally, we emphasize that when the element $y \in \mathcal{X}$ is a non-member for the language $\mathcal{L}$, then under a binding crs, the "**If**" condition in step 4 of $\mathsf{C}_{\mathsf{Ext,MMap}}$ is never satisfied. This follows from the perfect extractability guarantee of the NIZK proof system. However, the condition may be satisfied during some hybrid in the proof of security, under a hiding crs when the element $y \in \mathcal{X}$ is a member for the language $\mathcal{L}$.

Zero-Testing Encodings. Given the aforementioned extraction procedure, zero-testing an encoding at any given level is trivial. We simply apply the extraction procedure to the encoding, and check if the extracted group element g^* is equal to g^0 for any $g \in \mathbb{G}$.

5.6 The Overall Construction

We are now ready to formally summarize our construction of an SDDH-hard MMap.

- $\mathsf{Setup}(1^\lambda, N)$: On input the security parameter λ and the degree of multilinearity N, do the following:
  1. Sample (in binding mode) $(\mathsf{crs}, \mathsf{t}) \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$.
  2. Sample $(\mathsf{pk}_0, \mathsf{sk}_0), (\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathsf{FHE.Gen}(1^\lambda)$.
  3. Sample a non-member instance $y \leftarrow \mathcal{X} \setminus \mathcal{L}$.
  4. Compute $\mathsf{C}_{\mathsf{Add,piO}} = \mathsf{piO.Obf}(\mathsf{C}_{\mathsf{Add,MMap}})$, where $\mathsf{C}_{\mathsf{Add,MMap}}$ is as defined in Figure 15.
  5. Compute $\mathsf{C}_{\mathsf{Inv,piO}} = \mathsf{piO.Obf}(\mathsf{C}_{\mathsf{Inv,MMap}})$, where $\mathsf{C}_{\mathsf{Inv,MMap}}$ is as defined in Figure 17.
  6. Compute $\mathsf{C}_{\mathsf{Mult,piO}} = \mathsf{piO.Obf}(\mathsf{C}_{\mathsf{Mult,MMap}})$, where $\mathsf{C}_{\mathsf{Mult,MMap}}$ is as defined in Figure 19.
  7. Compute $\mathsf{C}_{\mathsf{Ext,piO}} = \mathsf{piO.Obf}(\mathsf{C}_{\mathsf{Ext,MMap}})$, where $\mathsf{C}_{\mathsf{Ext,MMap}}$ is as defined in Figure 20.

8. Output the public parameter pp, where

$$\mathsf{pp} = \left(\mathsf{crs}, \mathsf{pk}_0, \mathsf{pk}_1, \mathsf{C}_{\mathsf{Add,piO}}, \mathsf{C}_{\mathsf{Inv,piO}}, \mathsf{C}_{\mathsf{Mult,piO}}, \mathsf{C}_{\mathsf{Ext,piO}}, y\right).$$

- $\mathsf{Encode}(\mathsf{pp}, a, [\mathbf{i}])$: On input the public parameter pp, a plaintext element $a \in \mathbb{Z}_q$ and $i \in [N]$ such that $i \leq j \leq N$, proceed as follows:

  1. Generate the following ciphertexts:

  $$\mathsf{ct}_0 = \mathsf{FHE.Enc}(\mathsf{pk}_0, (a, 0, 0, \mathbf{i}); \mathsf{r}_0) \quad , \quad \mathsf{ct}_1 \leftarrow \mathsf{FHE.Enc}(\mathsf{pk}_1, (a, 0, 0, \mathbf{i}); \mathsf{r}_1),$$

  where r_0 and r_1 denote uniformly sampled random coins.

  2. Generate the following proof:

  $$\pi \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, \mathsf{st}, \mathsf{wit}),$$

  where the statement st and the witness wit correspond to the NP language L (as defined in Figure 13) and are given as:

  $$\mathsf{st} = (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, y) \quad , \quad \mathsf{wit} = (a, \mathsf{r}_0, \mathsf{r}_1).$$

  3. Finally, output the encoding

  $$[a]_{[\mathbf{i}]} := (\mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, \pi).$$

- $\mathsf{Add}(\mathsf{pp}, [a_1]_{[\mathbf{i}]}, [a_2]_{[\mathbf{i}]})$: On input the public parameter pp and a pair of encodings

$$[a_1]_{[\mathbf{i}]} = (\mathsf{ct}_{1,0}, \mathsf{ct}_{1,1}, \mathbf{i}, \pi_1) \quad , \quad [a_2]_{[\mathbf{i}]} = (\mathsf{ct}_{2,0}, \mathsf{ct}_{2,1}, \mathbf{i}, \pi_2),$$

output the encoding $[a_1 + a_2]_{[\mathbf{i}]} = (\mathsf{ct}^*_0, \mathsf{ct}^*_1, \mathbf{i}, \pi^*)$, where

$$(\mathsf{ct}^*_0, \mathsf{ct}^*_1, \mathbf{i}, \pi^*) = \mathsf{piOEval}(\mathsf{C}_{\mathsf{Add,MMap}}, ((\mathsf{ct}_{1,0}, \mathsf{ct}_{1,1}, \mathbf{i}, \pi_1), (\mathsf{ct}_{2,0}, \mathsf{ct}_{2,1}, \mathbf{i}, \pi_2), y)).$$

- $\mathsf{Inv}(\mathsf{pp}, [a]_{[\mathbf{i}]})$: On input the public parameter pp and an encoding $[[a]_{[\mathbf{i}]} = (\mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, \pi)$, output the encoding $[-a]_{[\mathbf{i}]} = (\mathsf{ct}^*_0, \mathsf{ct}^*_1, \mathbf{i}, \pi^*)$, where

$$(\mathsf{ct}^*_0, \mathsf{ct}^*_1, \mathbf{i}, \pi^*) = \mathsf{piOEval}(\mathsf{C}_{\mathsf{Inv,MMap}}, (\mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, \pi, y)).$$

- $\mathsf{Mult}(\mathsf{pp}, [a_1]_{[\mathbf{i}]}, [a_2]_{[\mathbf{i}_2]})$: On input the public parameter pp and a pair of encodings

$$[a_1]_{[\mathbf{i}]} = (\mathsf{ct}_{1,0}, \mathsf{ct}_{1,1}, \mathbf{i}, \pi_1), \quad , \quad [a_2]_{[\mathbf{i}_2]} = (\mathsf{ct}_{2,0}, \mathsf{ct}_{2,1}, \mathbf{i}_2, \pi_2),$$

output the encoding $[a_1.a_2]_{[\mathbf{i}_1+\mathbf{i}_2]} = (\mathsf{ct}^*_0, \mathsf{ct}^*_1, \mathbf{i}_1 + \mathbf{i}_2, \pi^*)$, where

$$(\mathsf{ct}^*_0, \mathsf{ct}^*_1, \mathbf{i}_1 + \mathbf{i}_2, \pi^*) = \mathsf{piOEval}(\mathsf{C}_{\mathsf{Mult,MMap}}, ((\mathsf{ct}_{1,0}, \mathsf{ct}_{1,1}, \mathbf{i}, \pi_1), (\mathsf{ct}_{2,0}, \mathsf{ct}_{2,1}, \mathbf{i}_2, \pi_2), y)).$$

- $\mathsf{Ext}(\mathsf{pp}, [a]_{[\mathbf{i}]})$: On input the public parameter pp and an encoding $[a]_{[\mathbf{i}]} = (\mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, \pi)$, output $g^* = \mathsf{piOEval}(\mathsf{C}_{\mathsf{Ext,MMap}}, ((\mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, \pi), y))$.

- $\mathsf{ZeroTest}(\mathsf{pp}, [a]_{[\mathbf{i}]})$: On input the public parameter pp and an encoding $[a]_{[\mathbf{i}]} = (\mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, \pi)$, output b^*, where

$$b^* = \begin{cases} \perp & \text{if } \mathsf{piOEval}(\mathsf{C}_{\mathsf{Ext,MMap}}, ((\mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, \pi), y)) = \perp, \\ 1 & \text{if } \mathsf{piOEval}(\mathsf{C}_{\mathsf{Ext,MMap}}, ((\mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, \pi), y)) = g^0 \text{ for any group element } g \in \mathbb{G}, \\ 0 & \text{otherwise.} \end{cases}$$

5.7 Proof of SDDH

We state and prove the following theorem:

Theorem 5.1. *Our MMap construction is SDDH-hard assuming: (a) that deciding subset membership with respect to $\mathcal{L} \subset \mathcal{X}$ is computationally hard, (b) that NIZK satisfies* crs*-indistinguishability, perfect simulation-extractability under a binding* crs *and perfect zero-knowledge under a hiding* crs*, (c) that piO satisfies probabilistic-iO security, (d) that FHE is IND-CPA secure, and (e) that the group $\mathbb{G}$ is SDDH-hard.*

We prove hardness of SDDH over our proposed MMap construction via a sequence of hybrids, summarized in Table 3. The crux of the proof lies in switching the plaintexts underlying the challenge encodings in the SDDH game to a special form such that:

- The challenge encodings constitute a valid SDDH instance over the MMap whenever the extraction circuit embeds a valid SDDH tuple over the group $\mathbb{G}$.
- The challenge encodings constitute a random invalid SDDH instance over the MMap whenever the extraction circuit embeds a random invalid SDDH tuple over the group $\mathbb{G}$.

We then show via a sequence of hybrids that these two cases are indistinguishable based on a sequence of hybrids that exploit the computational security properties of the underlying cryptoprimitives and the hardness of SDDH over the group $\mathbb{G}$. Table 3 presents an overview of the changes made to the circuits and/or the challenge encodings across the various hybrids. This is very similar to the proof strategy for our SXDH-hard MMap construction, and hence the details are avoided.

Hybrid-0. In this hybrid, the public parameter pp is generated as in the real scheme. As part of the SDDH challenge, the adversary is provided with encodings of α_0, α_1 (uniformly random in $\mathbb{Z}_q$) and $\alpha_0 \cdot \alpha_1$, generated as per the real encoding algorithm corresponding to some challenge level $\mathbf{i}$ (may be adversarially chosen).

Hybrid-1. In this hybrid, we change the public parameter pp as follows:

1. Switch crs for the NIZK proof system to hiding mode from binding mode.
2. Switch $y \in \mathcal{X}$ from a uniform non-member instance to a uniform member instance for the language $\mathcal{L}$ with witness wit_y.
3. Compute $\mathsf{C}_{\mathsf{Add},\mathsf{piO}} = \mathsf{piO}.\mathsf{Obf}(\widehat{\mathsf{C}}_{\mathsf{Add},\mathsf{MMap}})$, where $\widehat{\mathsf{C}}_{\mathsf{Add},\mathsf{MMap}}$ is as described in Figure 21.
4. Compute $\mathsf{C}_{\mathsf{Inv},\mathsf{piO}} = \mathsf{piO}.\mathsf{Obf}(\widehat{\mathsf{C}}_{\mathsf{Inv},\mathsf{MMap}})$, where $\widehat{\mathsf{C}}_{\mathsf{Inv},\mathsf{MMap}}$ is as described in Figure 22.
5. Compute $\mathsf{C}_{\mathsf{Mult},\mathsf{piO}} = \mathsf{piO}.\mathsf{Obf}(\widehat{\mathsf{C}}_{\mathsf{Mult},\mathsf{MMap}})$, where $\widehat{\mathsf{C}}_{\mathsf{Mult},\mathsf{MMap}}$ is as described in Figure 23.
6. Compute $\mathsf{C}_{\mathsf{Ext},\mathsf{piO}} = \mathsf{piO}.\mathsf{Obf}(\widehat{\mathsf{C}}_{\mathsf{Ext},\mathsf{MMap},1,0})$, where $\widehat{\mathsf{C}}_{\mathsf{Ext},\mathsf{MMap},1,0}$ is as described in Figure 24.

Lemma 5.2. *Hybrid-1 is computationally indistinguishable from Hybrid-0 assuming: (a) that deciding subset membership with respect to $\mathcal{L} \subset \mathcal{X}$ is computationally hard, (b) that NIZK satisfies* crs*-indistinguishability, perfect simulation-extractability under a binding* crs *and perfect zero-knowledge under a hiding* crs*, and (c) that piO satisfies probabilistic-iO security.*

The proof of this lemma is very similar to the proof of Lemma 4.2 and is hence not detailed.

Table 3: Overview of hybrids for proof of SDDH

Hybrid	crs	y	$\mathsf{C}_{\phi,\mathsf{MMap}}$ knows for $\phi \in \{\mathsf{Add}, \mathsf{Inv}, \mathsf{Mult}\}$	$\mathsf{C}_{\mathsf{Ext},\mathsf{MMap}}$ knows	(g, h, h') is of the form	Challenge FHE Ciphertexts (Set-1) are encryptions of	Challenge FHE Ciphertexts (Set-2) are encryptions of	Comments
0	binding	$\notin \mathcal{L}$	$(\mathsf{sk}_0, \mathsf{sk}_1, \mathsf{t})$	$(\mathsf{sk}_0, \mathsf{sk}_1, \mathsf{t})$	$(g, g^\gamma, g^{\gamma^*})$	$(\alpha, 0, 0)$ $(\alpha^2, 0, 0)$	$(\alpha, 0, 0)$ $(\alpha^2, 0, 0)$	—
1	hiding	$\in \mathcal{L}$	—	sk_1	$(g, g^\gamma, g^{\gamma^2})$	$(\alpha, 0, 0)$ $(\alpha^2, 0, 0)$	$(\alpha, 0, 0)$ $(\alpha^2, 0, 0)$	NIZK security + subspace-membership hardness + piO security
2	hiding	$\in \mathcal{L}$	—	sk_1	$(g, g^\gamma, g^{\gamma^2})$	$((\alpha - \mu \cdot \gamma), \mu, 0)$ $((\alpha - \mu \cdot \gamma)^2, (2 \cdot \alpha \cdot \mu), -(\mu)^2)$	$(\alpha, 0, 0)$ $(\alpha^2, 0, 0)$	FHE IND-CPA security
3	hiding	$\in \mathcal{L}$	—	sk_0	$(g, g^\gamma, g^{\gamma^2})$	$((\alpha - \mu \cdot \gamma), \mu, 0)$ $((\alpha - \mu \cdot \gamma)^2, (2 \cdot \alpha \cdot \mu), -(\mu)^2)$	$(\alpha, 0, 0)$ $(\alpha^2, 0, 0)$	NIZK security + piO security
4	hiding	$\in \mathcal{L}$	—	sk_0	$(g, g^\gamma, g^{\gamma^2})$	$((\alpha - \mu \cdot \gamma), \mu, 0)$ $((\alpha - \mu \cdot \gamma)^2, (2 \cdot \alpha \cdot \mu), -(\mu)^2)$	$((\alpha - \mu \cdot \gamma), \mu, 0)$ $((\alpha - \mu \cdot \gamma)^2, (2 \cdot \alpha \cdot \mu), -(\mu)^2)$	FHE IND-CPA security
5	hiding	$\in \mathcal{L}$	—	sk_0	$(g, g^\gamma, g^{\gamma^*})$	$((\alpha - \mu \cdot \gamma), \mu, 0)$ $((\alpha - \mu \cdot \gamma)^2, (2 \cdot \alpha \cdot \mu), -(\mu)^2)$	$((\alpha - \mu \cdot \gamma), \mu, 0)$ $((\alpha - \mu \cdot \gamma)^2, (2 \cdot \alpha \cdot \mu), -(\mu)^2)$	SDDH
6	hiding	$\in \mathcal{L}$	—	sk_0	$(g, g^\gamma, g^{\gamma^*})$	$((\alpha - \mu \cdot \gamma), \mu, 0)$ $((\alpha - \mu \cdot \gamma)^2, (2 \cdot \alpha \cdot \mu), -(\mu)^2)$	$(\alpha, 0, 0)$ $((\alpha^2 + \mu^2 \cdot (\gamma^2 - \gamma^*)), 0, 0)$	FHE IND-CPA security
7	hiding	$\in \mathcal{L}$	—	sk_1	$(g, g^\gamma, g^{\gamma^*})$	$((\alpha - \mu \cdot \gamma), \mu, 0)$ $((\alpha - \mu \cdot \gamma)^2, (2 \cdot \alpha \cdot \mu), -(\mu)^2)$	$(\alpha, 0, 0)$ $((\alpha^2 + \mu^2 \cdot (\gamma^2 - \gamma^*)), 0, 0)$	NIZK security + piO security
8	hiding	$\in \mathcal{L}$	—	sk_1	$(g, g^\gamma, g^{\gamma^*})$	$(\alpha, 0, 0)$ $((\alpha^2 + \mu^2 \cdot (\gamma^2 - \gamma^*)), 0, 0)$	$(\alpha, 0, 0)$ $((\alpha^2 + \mu^2 \cdot (\gamma^2 - \gamma^*)), 0, 0)$	FHE IND-CPA security
9	binding	$\notin \mathcal{L}$	$(g, g^\gamma, g^{\gamma^*})$	$(\mathsf{sk}_0, \mathsf{sk}_1, \mathsf{t})$	$(\mathsf{sk}_0, \mathsf{sk}_1, \mathsf{t})$	$(\alpha, 0, 0)$ $((\alpha^2 + \mu^2 \cdot (\gamma^2 - \gamma^*)), 0, 0)$	$(\alpha, 0, 0)$ $((\alpha^2 + \mu^2 \cdot (\gamma^2 - \gamma^*)), 0, 0)$	NIZK security + subspace-membership hardness + piO security

Hybrid-2. In this hybrid, we change the challenge encodings. Suppose that the SDDH instance (g, h, h') in the extraction circuit $\widehat{C}_{\mathsf{Ext},\mathsf{MMap},1,0}$ is distributed as:

$$(g, h, h') = (g, g^\gamma, g^{\gamma^2}).$$

We switch the plaintext underlying the *first* FHE ciphertext in each challenge encoding as follows (where μ_0 and μ_1 are sampled uniformly from $\mathbb{Z}_q^*$):

$$\begin{aligned} (\alpha, 0, 0, \mathbf{i}) &\rightarrow ((\alpha - \mu \cdot \gamma), \mu, 0, \mathbf{i}) \\ (\alpha^2, 0, 0, \mathbf{i}) &\rightarrow ((\alpha - \mu \cdot \gamma)^2, (2 \cdot \alpha \cdot \mu), -(\mu)^2, \mathbf{i}) \end{aligned}$$

Lemma 5.3. *Hybrid-2 is computationally indistinguishable from Hybrid-1 assuming that FHE is IND-CPA secure with respect to the public key-secret key pair* $(\mathsf{pk}_0, \mathsf{sk}_0)$.

The proof of this lemma is very similar to the proof of Lemma 4.3 and is hence not detailed.

Hybrid-3. In this hybrid, we change the public parameter pp as follows: compute $\mathsf{C}_{\mathsf{Ext},\mathsf{piO}} = \mathsf{piO.Obf}(\widehat{\mathsf{C}}_{\mathsf{Ext},\mathsf{MMap},0,0})$, where $\widehat{\mathsf{C}}_{\mathsf{Ext},\mathsf{MMap},0,0}$ is as defined in Figure 24.

Inputs: A tuple of the form: $((\mathsf{ct}_{1,0}, \mathsf{ct}_{1,1}, \mathbf{i}, \pi_1), (\mathsf{ct}_{2,0}, \mathsf{ct}_{2,1}, \mathbf{i}, \pi_2), y \in \mathcal{X})$, where the proofs π_1 and π_2 are over the language L.

Outputs: Either a tuple of the form $(\mathsf{ct}_0^*, \mathsf{ct}_1^*, \mathbf{i}, \pi^*)$, or $\perp$.

Computation: Proceed as follows:

1. **If** any of the following is true:
   (a) $\mathsf{NIZK.Verify}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_{1,0}, \mathsf{ct}_{1,1}, \mathbf{i}, y \in \mathcal{X}), \pi_1) = 0$.
   (b) $\mathsf{NIZK.Verify}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_{2,0}, \mathsf{ct}_{2,1}, \mathbf{i}, y \in \mathcal{X}), \pi_2) = 0$.
   then output $\perp$.
2. // Decryption omitted — avoids dependency on sk_0 and sk_1.
3. Compute the following:
$$\mathsf{ct}_0^* = \mathsf{FHE.Eval}(\mathsf{pk}_0, \mathsf{ct}_{1,0}, \mathsf{ct}_{2,0}, \mathsf{C}_{\mathsf{Add,FHE}}) \quad , \quad \mathsf{ct}_1^* = \mathsf{FHE.Eval}(\mathsf{pk}_1, \mathsf{ct}_{1,1}, \mathsf{ct}_{2,1}, \mathsf{C}_{\mathsf{Add,FHE}}).$$
4. // If check omitted — avoids dependency on sk_0 and sk_1.
   Proceed as follows:
   (a) // Extraction omitted — avoids dependency on extraction trapdoor t.
   (b) // Check omitted.
   (c) Generate $\pi^* \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0^*, \mathsf{ct}_1^*, \mathbf{i}, y), \mathsf{wit}_y)$.
5. // Proof generation omitted — avoids dependency on sk_0 and sk_1.
6. Output $(\mathsf{ct}_0^*, \mathsf{ct}_1^*, \mathbf{i}, \pi^*)$.

Figure 21: Let $\mathsf{crs} \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$ be in hiding mode. Additionally, let $(\mathsf{pk}_0, \mathsf{sk}_0), (\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathsf{FHE.Gen}(1^\lambda)$. Also, for $y \in \mathcal{L}$, let wit_y be a membership-witness such that $(y, \mathsf{wit}_y) \in \mathsf{R}_{\mathcal{L}}$. We define a probabilistic circuit $\widehat{\mathsf{C}}_{\mathsf{Add,MMap}}$ (which has only the public keys pk_0 and pk_1 hard-wired into it) as above.

Inputs: A tuple of the form: $(\{\mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, \pi, y \in \mathcal{X})$, where the proof π is over the language L.

Outputs: Either a tuple of the form $(\mathsf{ct}_0^*, \mathsf{ct}_1^*, \mathbf{i}, \pi^*)$, or $\perp$.

Computation: Proceed as follows:

1. If $\mathsf{NIZK.Verify}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, y), \pi) = 0$, then outut $\perp$.
2. // Decryption omitted — avoids dependency on sk_0 and sk_1.
3. Compute the following:
$$\mathsf{ct}_0^* = \mathsf{FHE.Eval}(\mathsf{pk}_0, \mathsf{ct}_0, \mathsf{C}_{\mathsf{Inv,FHE}}) \quad , \quad \mathsf{ct}_1^* = \mathsf{FHE.Eval}(\mathsf{pk}_1, \mathsf{ct}_1, \mathsf{C}_{\mathsf{Inv,FHE}}).$$
4. // If check omitted — avoids dependency on sk_0 and sk_1.
   Proceed as follows:
   (a) // Extraction omitted — avoids dependency on extraction trapdoor t.
   (b) // Check omitted.
   (c) Generate $\pi^* \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0^*, \mathsf{ct}_1^*, \mathbf{i}, y), \mathsf{wit}_y)$.
5. // Proof generation omitted — avoids dependency on sk_0 and sk_1.
6. Output $(\mathsf{ct}_0^*, \mathsf{ct}_1^*, \mathbf{i}, \pi^*)$.

Figure 22: Next, Let $\mathsf{crs} \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$ be in hiding mode. Additionally, let $(\mathsf{pk}_0, \mathsf{sk}_0), (\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathsf{FHE.Gen}(1^\lambda)$. Also, for $y \in \mathcal{L}$, let wit_y be a membership-witness such that $(y, \mathsf{wit}_y) \in \mathsf{R}_{\mathcal{L}}$. We define a probabilistic circuit $\widehat{\mathsf{C}}_{\mathsf{Inv,MMap}}$ (which has only the public keys pk_0 and pk_1 hard-wired into it) as above

Lemma 5.4. *Hybrid-3 is computationally indistinguishable from Hybrid-2 assuming: (a) that NIZK satisfies* crs*-indistinguishability, perfect simulation-extractability under a binding* crs *and perfect zero-knowledge under a hiding* crs*, and (b) that piO satisfies probabilistic-iO security.*

The proof of this lemma is very similar to the proof of Lemma 4.4 and is hence not detailed.

Inputs: A tuple of the form: $((\mathsf{ct}_{1,0}, \mathsf{ct}_{1,1}, \mathbf{i}_1, \pi_1), (\mathsf{ct}_{2,0}, \mathsf{ct}_{2,1}, \mathbf{i}_2, \pi_2), y \in \mathcal{X})$, where the proofs π_1 and π_2 are over the language L.

Outputs: Either a tuple of the form $(\mathsf{ct}_0^*, \mathsf{ct}_1^*, i, k, \pi^*)$, or $\perp$.

Computation: Proceed as follows:

1. **If** any of the following is true:
   (a) $\mathsf{NIZK.Verify}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_{1,0}, \mathsf{ct}_{1,1}, \mathbf{i}_1, y \in \mathcal{X}), \pi_1) = 0$.
   (b) $\mathsf{NIZK.Verify}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_{2,0}, \mathsf{ct}_{2,1}, \mathbf{i}_2, y \in \mathcal{X}), \pi_2) = 0$.
   then output $\perp$.
2. // Decryption omitted — avoids dependency on sk_0 and sk_1.
3. Compute the following:
$$\mathsf{ct}_0^* = \mathsf{FHE.Eval}(\mathsf{pk}_0, \mathsf{ct}_{1,0}, \mathsf{ct}_{2,0}, \mathsf{C}_{\mathsf{Mult,FHE}}) \quad , \quad \mathsf{ct}_1^* = \mathsf{FHE.Eval}(\mathsf{pk}_1, \mathsf{ct}_{1,1}, \mathsf{ct}_{2,1}, \mathsf{C}_{\mathsf{Mult,FHE}}).$$
4. // If check omitted — avoids dependency on sk_0 and sk_1.
   Proceed as follows:
   (a) // Extraction omitted — avoids dependency on extraction trapdoor t.
   (b) // Check omitted.
   (c) Generate $\pi^* \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0^*, \mathsf{ct}_1^*, \mathbf{i}_1 + \mathbf{i}_2, y), \mathsf{wit}_y)$.
5. // Proof generation omitted — avoids dependency on sk_0 and sk_1.
6. Output $(\mathsf{ct}_0^*, \mathsf{ct}_1^*, \mathbf{i}_1 + \mathbf{i}_2, \pi^*)$.

Figure 23: Let $\mathsf{crs} \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$ be in hiding mode. Additionally, let $(\mathsf{pk}_0, \mathsf{sk}_0), (\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathsf{FHE.Gen}(1^\lambda)$. Also, for $y \in \mathcal{L}$, let wit_y be a membership-witness such that $(y, \mathsf{wit}_y) \in \mathsf{R}_{\mathcal{L}}$. For each $b^* \in \{0, 1\}$, we define a probabilistic circuit $\widehat{\mathsf{C}}_{\mathsf{Mult,MMap}}$ (which has only the public keys pk_0 and pk_1 hard-wired into it) as above.

Inputs: A tuple of the form: $((\mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, \pi), y \in \mathcal{X})$, where the proof π is over the language L.

Outputs: Either a group element g^* or $\perp$.

Computation: Proceed as follows:

1. If $\mathsf{NIZK.Verify}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, y), \pi) = 0$, then output $\perp$.
2. Recover the following: $(\{a_{\beta,\ell}\}_{\ell \in [0,3]}, \mathbf{i}_\beta) = \mathsf{FHE.Dec}(\mathsf{sk}_\beta, \mathsf{ct}_\beta)$.
   // Decryption using $\mathsf{sk}_{1-\beta}$ omitted.
3. Compute $g^* = g^{a_{\beta,0}} \cdot h^{a_{\beta,1}} \cdot (h')^{a_{\beta,2}}$.
4. // If clause omitted — avoids dependency on $\mathsf{sk}_{1-\beta}$ and extraction trapdoor t.
5. Output g^*.

Figure 24: Let $\mathsf{crs} \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$ be in hiding mode and let $(\mathsf{pk}_0, \mathsf{sk}_0), (\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathsf{FHE.Gen}(1^\lambda)$. Also, let $g, h, h' \in \mathbb{G}$ be group elements that are distributed as per $\mathcal{D}_{\mathsf{SDDH},\mathbb{G}}$ if $\beta' = 0$ and uniformly randomly sampled if $\beta' = 1$, subject to the restriction that the tuple always uses the same base element g. For each $\beta, \beta' \in \{0, 1\}$, we define a deterministic circuit $\widehat{C}_{\mathsf{Ext,MMap},\beta,\beta'}$ (which has the public keys pk_0 and pk_1, the secret key sk_β and the collection of group elements (g, h, h') hard-wired into it) as in Figure 24 above.

Hybrid-4. In this hybrid, we change the challenge encodings. Again, suppose that the SDDH instance (g, h, h') in the extraction circuit $\widehat{C}_{\mathsf{Ext,MMap},0,0}$ is distributed as:

$$(g, h, h') = (g, g^\gamma, g^{\gamma^2}).$$

Switch the plaintext underlying the *second* FHE ciphertext in each challenge encoding as follows (where μ_0 and μ_1 are sampled uniformly from $\mathbb{Z}_q^*$):

$$\begin{aligned} (\alpha, 0, 0, \mathbf{i}) &\rightarrow ((\alpha - \mu \cdot \gamma), \mu, 0, \mathbf{i}) \\ (\alpha^2, 0, 0, \mathbf{i}) &\rightarrow ((\alpha - \mu \cdot \gamma)^2, (2 \cdot \alpha \cdot \mu), -(\mu)^2, \mathbf{i}) \end{aligned}$$

Lemma 5.5. *Hybrid-4 is computationally indistinguishable from Hybrid-3 assuming that FHE is IND-CPA secure with respect to the public key-secret key pair* $(\mathsf{pk}_1, \mathsf{sk}_1)$.

The proof of this lemma is very similar to the proof of Lemma 4.5 and is hence not detailed.

Hybrid-5. In this hybrid, we change the public parameter pp as follows: compute $\mathsf{C}_{\mathsf{Ext,piO}} = \mathsf{piO.Obf}(\widehat{\mathsf{C}}_{\mathsf{Ext,MMap},0,1})$, where $\widehat{\mathsf{C}}_{\mathsf{Ext,MMap},0,1}$ is as defined in Figure 24.

Lemma 5.6. *Hybrid-5 is computationally indistinguishable from Hybrid-4 assuming that the SDDH assumption holds over the group* $\mathbb{G}$.

Proof. Note that the only difference between hybrids 4 and 5 is the manner in which the obfuscated extraction circuit $\mathsf{C}_{\mathsf{Ext,piO}}$ is generated as part of the public parameter.

In Hybrid-4, $\mathsf{C}_{\mathsf{Ext,piO}}$ is an obfuscation of the circuit $\widehat{\mathsf{C}}_{\mathsf{Ext,MMap},0,0}$, while in Hybrid-5, $\mathsf{C}_{\mathsf{Ext,piO}}$ is an obfuscation of the circuit $\widehat{\mathsf{C}}_{\mathsf{Ext,MMap},0,1}$ (where both circuits are as defined in Figure 12).

Now, observe that the only difference between the extraction circuits $\widehat{\mathsf{C}}_{\mathsf{Ext,MMap},0,0}$ and $\widehat{\mathsf{C}}_{\mathsf{Ext,MMap},0,1}$ is that the former embeds a valid SDDH tuple over the group $\mathbb{G}$ while the latter embeds a uniformly random three-tuple of group elements over the group $\mathbb{G}$.

It immediately follows that Hybrid-5 is computationally indistinguishable from Hybrid-4 assuming that the SDDH assumption holds over the group $\mathbb{G}$.

Hybrid-6. In this hybrid, we change the challenge encodings. Recall that in hybrid-5, (g, h, h') in the extraction circuit $\widehat{C}_{\mathsf{Ext,MMap},0,1}$ is distributed as:

$$(g, h, h') = (g, g^{\gamma}, g^{\gamma^*}).$$

Switch back the plaintext underlying the *second* FHE ciphertext in each challenge encoding as follows (where μ_0 and μ_1 are sampled uniformly from $\mathbb{Z}_q^*$):

$$\begin{aligned} ((\alpha - \mu \cdot \gamma), \mu, 0, \mathbf{i}) &\rightarrow (\alpha, 0, 0, \mathbf{i}) \\ ((\alpha - \mu \cdot \gamma)^2, (2 \cdot \alpha \cdot \mu), -(\mu)^2, \mathbf{i}) &\rightarrow ((\alpha^2 + \mu^2 \cdot (\gamma^2 - \gamma^*)), 0, 0, \mathbf{i}) \end{aligned}$$

Lemma 5.7. *Hybrid-6 is computationally indistinguishable from Hybrid-5 assuming that FHE is IND-CPA secure with respect to the public key-secret key pair* $(\mathsf{pk}_1, \mathsf{sk}_1)$.

The proof of this lemma is very similar to the proof of Lemma 4.7 and is hence not detailed.

Hybrid-7. In this hybrid, we change the public parameter pp as follows: compute $\mathsf{C}_{\mathsf{Ext,piO}} = \mathsf{piO.Obf}(\widehat{\mathsf{C}}_{\mathsf{Ext,MMap},1,1})$, where $\widehat{\mathsf{C}}_{\mathsf{Ext,MMap},1,1}$ is as defined in Figure 24.

Lemma 5.8. *Hybrid-7 is computationally indistinguishable from Hybrid-6 assuming: (a) that NIZK satisfies* crs*-indistinguishability, perfect simulation-extractability under a binding* crs *and perfect zero-knowledge under a hiding* crs*, and (b) that piO satisfies probabilistic-iO security.*

The proof of this lemma is very similar to the proof of Lemma 4.8 and is hence not detailed.

Hybrid-8. In this hybrid, we change the challenge encodings. Recall that in hybrid-7, (g, h, h') in the extraction circuit $\widehat{C}_{\mathsf{Ext,MMap},1,1}$ is distributed as:

$$(g, h, h') = (g, g^{\gamma}, g^{\gamma^*}).$$

Switch back the plaintext underlying the *first* FHE ciphertext in each challenge encoding as follows (where μ_0 and μ_1 are sampled uniformly from $\mathbb{Z}_q^*$):

$$\begin{aligned} ((\alpha - \mu \cdot \gamma), \mu, 0, \mathbf{i}) &\rightarrow (\alpha, 0, 0, \mathbf{i}) \\ ((\alpha - \mu \cdot \gamma)^2, (2 \cdot \alpha \cdot \mu), -(\mu)^2, \mathbf{i}) &\rightarrow ((\alpha^2 + \mu^2 \cdot (\gamma^2 - \gamma^*)), 0, 0, \mathbf{i}) \end{aligned}$$

Lemma 5.9. *Hybrid-8 is computationally indistinguishable from Hybrid-7 assuming that FHE is IND-CPA secure with respect to the public key-secret key pair* $(\mathsf{pk}_0, \mathsf{sk}_0)$.

The proof of this lemma is very similar to the proof of Lemma 4.9 and is hence not detailed.

Hybrid-9. In this hybrid, we change the public parameter pp as follows:

1. Switch crs for the NIZK proof system to binding mode, as in the real scheme
2. Switch $y \in \mathcal{X}$ to a uniform non-member instance, as in the real scheme.
3. Compute $\mathsf{C}_{\mathsf{Add,piO}} = \mathsf{piO.Obf}(\mathsf{C}_{\mathsf{Add,MMap}})$, as in the real scheme.
4. Compute $\mathsf{C}_{\mathsf{Inv,piO}} = \mathsf{piO.Obf}(\mathsf{C}_{\mathsf{Inv,MMap}})$, as in the real scheme.
5. Compute $\mathsf{C}_{\mathsf{Mult,piO}} = \mathsf{piO.Obf}(\mathsf{C}_{\mathsf{Mult,MMap}})$, as in the real scheme.
6. Compute $\mathsf{C}_{\mathsf{Ext,piO}} = \mathsf{piO.Obf}(\mathsf{C}_{\mathsf{Ext,MMap}})$, as in the real scheme.

Lemma 5.10. *Hybrid-9 is computationally indistinguishable from Hybrid-8 assuming: (a) that deciding subset membership with respect to $\mathcal{L} \subset \mathcal{X}$ is computationally hard, (b) that NIZK satisfies* crs*-indistinguishability, perfect simulation-extractability under a binding* crs *and perfect zero-knowledge under a hiding* crs*, and (c) that piO satisfies probabilistic-iO security.*

The proof of this lemma is very similar to the proof of Lemma 4.10 and is hence not detailed.

Finally, Hybrid-9 is statistically indistinguishable from the game where the adversary is given a random invalid SDDH instance generated as per the real scheme. This completes the proof of Theorem 5.1.

6 Constructing an EDDH-Hard Symmetric MMap

We now show how to extend the approach used for our asymmetric MMap constructions in Sections 4 and 5 to construct an exponent-DDH (EDDH)-hard *symmetric MMap*. The key challenge when extending our approach to the symmetric MMap setting is to take into account the fact that the adversary can multiply encodings at the same level. We provide some more intuition below.

Recall that the $(N+1)$-EDDH assumption over a degree-N symmetric MMap requires that the following indistinguishability holds for any level $i \in [N]$:

$$\left([\alpha]_i, [\alpha^{N+1}]_i\right) \quad \stackrel{c}{\approx} \quad \left([\alpha]_i, [\alpha^*]_i\right),$$

where $\alpha, \alpha^* \leftarrow \mathbb{Z}_q$. Note that the description of the assumption captures only a pair of encodings that an $(N+1)$-EDDH adversary sees. However, in the symmetric MMap setting, the adversary can actually create (via multiplication) all encodings of one of the following forms:

- $\left\{\left[\alpha^{\ell+(N+1)\ell'}\right]\right\}_{(\ell+\ell')\in[0,N]}$ when provided with a valid $(N+1)$-EDDH instance.
- $\left\{\left[\alpha^{\ell}\cdot(\alpha^*)^{\ell'}\right]\right\}_{(\ell+\ell')\in[0,N]}$ when provided with a random invalid $(N+1)$-EDDH instance.

This in turn implies that when we prove $(N+1)$-EDDH over our symmetric MMap construction, the challenger should be able to simulate the encodings for each of the aforementioned "cross-product" terms. Note that this issue is not encountered in the asymmetric setting, where the adversary is inherently restricted from multiplying a challenge encoding with itself or other challenge encodings at the same level.

We get around this issue as follows: our symmetric MMap construction uses a pairing-free group $\mathbb{G}$ of prime order q, such that letting $g \leftarrow \mathbb{G}$ and $\gamma, \delta \leftarrow \mathbb{Z}_q^*$, the following indistinguishability holds for any $N = \mathrm{poly}(\lambda)$ (λ being the security parameter):

$$\left(\left\{g^{\gamma^{\ell+(N+1)\ell'}}\right\}_{(\ell+\ell')\in[0,N]}\right) \overset{c}{\approx} \left(\left\{g^{\gamma^{\ell}\cdot\delta^{\ell'}}\right\}_{(\ell+\ell')\in[0,N]}\right).$$

We let $\mathcal{D}_{\text{strong}-(N+1)-\text{EDDH},\mathbb{G}}$ and $\mathcal{D}'_{\text{strong}-(N+1)-\text{EDDH},\mathbb{G}}$ denote the left and right distributions in the aforementioned expression.

We refer to this indistinguishability assumption as the "strong"-$(N+1)$-EDDH assumption over the group $\mathbb{G}$. We show later that the "strong"-$(N+1)$-EDDH assumption for $N \geq 2$ is implied by the DDH assumption.

Note that this assumption contains all the "cross-terms" that the challenger would need to simulate for the adversary in the $(N+1)$-EDDH game with respect to the MMap. Intuitively, this is what allows our proof of $(N+1)$-EDDH to go through. We expand more on this subsequently.

To summarize, we show a how to construct an $(N+1)$-EDDH-hard symmetric MMap given the following cryptoprimitives:

- A probabilistic-iO scheme $\mathsf{piO} = (\mathsf{piO.Obf}, \mathsf{piOEval})$.
- A fully-homomorphic encryption scheme $\mathsf{FHE} = (\mathsf{FHE.Gen}, \mathsf{FHE.Enc}, \mathsf{FHE.Dec}, \mathsf{FHE.Eval})$ such that the message space is $\mathbb{Z}_q$ for some prime $q = \mathrm{poly}(\lambda)$ (λ being the security parameter).
- A simulation-extractable $\mathsf{NIZK} = (\mathsf{NIZK.Setup}, \mathsf{NIZK.Prove}, \mathsf{NIZK.Verify})$.
- A pair of sets $(\mathcal{X}, \mathcal{L})$ such that $\mathcal{L} \subset \mathcal{X}$ and:
  1. Given $x \in \mathcal{X}$ it is *computationally hard* to decide if $x \in \mathcal{L}$.
  2. For each $y \in \mathcal{L}$, there exists a *unique* witness wit_y for the statement $y \in \mathcal{L}$.

  We define a relation $\mathsf{R}_{\mathcal{L}}$ as follows: $(y, \mathsf{wit}_y) \in R_{\mathcal{L}}$ if and only if wit_y is a witness for the statement $y \in \mathcal{L}$.
- A strong-$(N+1)$-EDDH-hard pairing-free group $\mathbb{G}$.

6.1 Encodings

Level-Zero Encodings. As in the previous constructions, level-zero encodings are treated slightly different from encodings at other "non-zero" levels. In particular, our level-zero encodings are equipped with their own set of algorithms for encoding, manipulation, extraction and zero-testing. We informally mention these for the sake of completeness.

The level-zero encoding of a plaintext element $a \in \mathbb{Z}_q$ is a itself. Adding/multiplying two level-zero encodings (equivalently, additively inverting a level-zero encoding) is simply done via addition/multiplication (equivalently, additive inversion) in $\mathbb{Z}_q$. Multiplying a level-zero encoding with any other encoding at some level i should result in an encoding at level-i. This is implemented with a shift-and-add algorithm built on top of the standard encoding addition algorithm described subsequently in Section 6.2. Finally, extracting a level-zero encoding $a \in \mathbb{Z}_q$ outputs g^a, where g

is a fixed generator for the group $\mathbb{G}$. As will be clear later, this is consistent with the extraction algorithm for any other level i. Zero-testing follows trivially.

We again omit formal descriptions of operations for level-0 encodings to ease notation and focus on the more interesting cases at higher encoding levels.

Level-i Encodings. We now describe the procedure of encoding a plaintext element at any interval level. An encoding of a plaintext element $a \in \mathbb{Z}_q$ at level $i \leq N$ is a tuple of the form: $(\mathsf{ct}_0, \mathsf{ct}_1, i, \pi)$, where:

- ct_0 and ct_1 are FHE encryptions of the tuple

$$\left(\{a_{\ell,\ell'}\}_{(\ell+\ell')\in[0,N]}, i\right),$$

  under the public key-secret key pairs $(\mathsf{pk}_0, \mathsf{sk}_0)$ and $(\mathsf{pk}_1, \mathsf{sk}_1)$ respectively, subject to the restriction that for each $(\ell, \ell') \neq (0, 0)$, we have $a_{\ell,\ell'} = 0$.

- π is a NIZK proof for the statement $\mathsf{st} = (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0, \mathsf{ct}_1, i, y \in \mathcal{X})$ under the NP language L described in Figure 25, where $y \in \mathcal{X}$ is available as part of the public parameter for our MMap scheme.

Note that the additional zero entries encrypted in each encoding appear redundant, but will be useful later in the proof of security.

Language L:

Statement: The statement st is as follows: $\mathsf{st} = (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0, \mathsf{ct}_1, i, y \in \mathcal{X})$

Witness: The witness has one of the following three forms:

1. $\mathsf{wit} = (\mathsf{sk}_0, \mathsf{sk}_1)$
2. $\mathsf{wit} = (\mathsf{m}, \mathsf{r}_0, \mathsf{r}_1)$
3. $\mathsf{wit} = \mathsf{wit}_y$

Relation: $\mathsf{R}(\mathsf{st}, \mathsf{wit}) = 1$ if and only if:

- **Either** $\mathsf{wit} = (\mathsf{sk}_0, \mathsf{sk}_1)$ and we have:
  1. $(\mathsf{pk}_0, \mathsf{sk}_0), (\mathsf{pk}_1, \mathsf{sk}_1) \in \mathcal{K}_{\mathrm{FHE}}$.
  2. $\mathsf{FHE.Dec}(\mathsf{sk}_0, \mathsf{ct}_0) = \mathsf{FHE.Dec}(\mathsf{sk}_1, \mathsf{ct}_1) = \left(\mathsf{m}, \{a_{\ell,\ell'}\}_{(\ell+\ell')\in[0,N]}, i\right)$ for some $\mathsf{m} \in \mathbb{Z}_q$, such that $a_{\ell,\ell'} = 0$ for each $(\ell, \ell') \neq (0, 0)$.
- **Or** $\mathsf{wit} = (\mathsf{m}, \mathsf{r}_0, \mathsf{r}_1)$ and we have:
  1. $\mathsf{m} \in \mathbb{Z}_q$.
  2. For each $b \in \{0, 1\}$, we have $\mathsf{ct}_b = \mathsf{FHE.Enc}\left(\mathsf{pk}_b, \left(\mathsf{m}, \{a_{\ell,\ell'}\}_{(\ell+\ell')\in[0,N]}, i\right); \mathsf{r}_b\right)$, such that $a_{\ell,\ell'} = 0$ for each $(\ell, \ell') \neq (0, 0)$.
- **Or** $\mathsf{wit} = \mathsf{wit}_y$ and we have: $(y, \mathsf{wit}_y) \in \mathsf{R}_{\mathcal{L}}$.

Figure 25: Let $\mathcal{K}_{\mathrm{FHE}}$ be the set of all valid key pairs under the FHE scheme. We define an NP language L as in the figure above. For simplicity, L parameterized by N – the number of levels for our MMap scheme.

6.2 Addition of Encodings

We now describe the procedure for adding two encodings at the same level. Suppose we have two encodings at the same level i of the form:

$$(\mathsf{ct}_{1,0}, \mathsf{ct}_{1,1}, i, \pi_1), (\mathsf{ct}_{2,0}, \mathsf{ct}_{2,1}, i, \pi_2).$$

Conceptually, we add these two encodings by exploiting the fully-homomorphic nature of the encryption scheme. More concretely, we homomorphically evaluate the circuit $\mathsf{C}_{\mathsf{Add,FHE}}$ (described in Figure 26) on the corresponding ciphertext components of the two input encodings to generate the ciphertext components for the output encoding. We also generate a proof for the validity of the resulting encoding using the tuple of secret keys $(\mathsf{sk}_0, \mathsf{sk}_1)$ as witness.

Inputs: A tuple of the form $((\{a_{\ell,\ell',1}\}_{(\ell+\ell')\in[0,N]}, i_1), (\{a_{\ell,\ell',2}\}_{(\ell+\ell')\in[0,N]}, i_2))$.

Computation: Proceed as follows:

1. If $i_1 \neq i_2$, output $\perp$.
2. Else, output $(\{a_{\ell,\ell',1} + a_{\ell,\ell',2}\}_{(\ell+\ell')\in[0,N]}, i_1)$.

Figure 26: Circuit $\mathsf{C}_{\mathsf{Add,FHE}}$

For technical reasons that are relevant to the proof of security, we check: (a) the validity of the proofs π_1 and π_2 for the language L that are provided as part of the input encodings, and (b) the "consistency" of the FHE ciphertexts that are provided as part of the input encodings, i.e., whether the ciphertexts encrypt the same underlying plaintext, and whether the plaintext is formatted as per the specifications of the scheme. While "validity" is publicly verifiable, verifying "consistency" required knowledge of the secret keys sk_0 and sk_1.

Figure 27 details the operation of the encoding-addition circuit $\mathsf{C}_{\mathsf{Add,MMap}}$. Of course, it embeds multiple secrets, including the secret keys sk_0 and sk_1, as well as the extraction trapdoor t for the NIZK. Hence, the circuit is not made public as is; we only make available an obfuscated version of the circuit obtained by running the evaluation algorithm of the probabilistic iO scheme piO on it.

We emphasize that when the element $y \in \mathcal{X}$ is a non-member for the language $\mathcal{L}$, then under a binding crs, the "**If**" condition in step 4 of $\mathsf{C}_{\mathsf{Add,MMap}}$ is never satisfied. This follows from the perfect extractability guarantee of the NIZK proof system. However, the condition may be satisfied during some hybrid in the proof of security, under a hiding crs when the element $y \in \mathcal{X}$ is a member for the language $\mathcal{L}$.

6.3 Inversion of Encodings

We now describe the procedure for additively inverting an encoding at any given level. Suppose we have an encoding at the level i of the form: $(\mathsf{ct}_0, \mathsf{ct}_1, i, \pi)$. Again, conceptually, we additively invert this encoding by exploiting the fully-homomorphic nature of the encryption scheme. More concretely, we homomorphically evaluate the circuit $\mathsf{C}_{\mathsf{Inv,FHE}}$ (described in Figure 28) on the corresponding ciphertext components of the input encoding to generate the ciphertext components for the output encoding. We also generate a proof for the validity of the resulting encoding using the tuple of secret keys $(\mathsf{sk}_0, \mathsf{sk}_1)$ as witness.

Similar to the addition procedure, for technical reasons that are relevant to the proof of security, we check the validity and consistency of the input encodings. Figure 29 details the operation of the encoding-inversion circuit $\mathsf{C}_{\mathsf{Inv,MMap}}$. Since it embeds the same set of secrets as the addition circuit, the circuit $\mathsf{C}_{\mathsf{Inv,MMap}}$ is not made public as is; we only make available an obfuscated version of the circuit obtained by running the evaluation algorithm of the probabilistic iO scheme piO on it.

Finally, we emphasize that when the element $y \in \mathcal{X}$ is a non-member for the language $\mathcal{L}$, then under a binding crs, the "**If**" condition in step 4 of $\mathsf{C}_{\mathsf{Add,MMap}}$ is never satisfied. This follows from the perfect extractability guarantee of the NIZK proof system. However, the condition may be satisfied during some hybrid in the proof of security, under a hiding crs when the element $y \in \mathcal{X}$ is a member for the language $\mathcal{L}$.

6.4 Multiplication of Encodings

We now describe the procedure for multiplying two encodings at levels i_1 and i_2 such that $i_1 + i_2 \leq N$:

$$(\mathsf{ct}_{1,0}, \mathsf{ct}_{1,1}, i_1, \pi_1), (\mathsf{ct}_{2,0}, \mathsf{ct}_{2,1}, i_2, \pi_2).$$

Encoding Addition Circuit: $\mathsf{C}_{\mathsf{Add,MMap}}$

Inputs: A tuple of the form: $((\mathsf{ct}_{1,0}, \mathsf{ct}_{1,1}, i, \pi_1), (\mathsf{ct}_{2,0}, \mathsf{ct}_{2,1}, i, \pi_2), y \in \mathcal{X})$, where the proofs π_1 and π_2 are over the language L.

Outputs: Either a tuple of the form $(\mathsf{ct}_0^*, \mathsf{ct}_1^*, i, \pi^*)$, or $\perp$.

Computation: Proceed as follows:

1. **If** any of the following is true:
   (a) $\mathsf{NIZK.Verify}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_{1,0}, \mathsf{ct}_{1,1}, i, y \in \mathcal{X}), \pi_1) = 0$.
   (b) $\mathsf{NIZK.Verify}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_{2,0}, \mathsf{ct}_{2,1}, i, y \in \mathcal{X}), \pi_2) = 0$.
   then output $\perp$.
2. Recover the following:
$$(\{a_{\ell,\ell',1,0}\}_{(\ell+\ell')\in[0,N]}, i_{1,0}) = \mathsf{FHE.Dec}(\mathsf{sk}_0, \mathsf{ct}_{1,0}), (\{a_{\ell,\ell',1,1}\}_{(\ell+\ell')\in[0,N]}, i_{1,1}) = \mathsf{FHE.Dec}(\mathsf{sk}_1, \mathsf{ct}_{1,1}),$$
$$(\{a_{\ell,\ell',2,0}\}_{(\ell+\ell')\in[0,N]}, i_{2,0}) = \mathsf{FHE.Dec}(\mathsf{sk}_0, \mathsf{ct}_{2,0}), (\{a_{\ell,\ell',2,1}\}_{(\ell+\ell')\in[0,N]}, i_{2,1}) = \mathsf{FHE.Dec}(\mathsf{sk}_1, \mathsf{ct}_{2,1}).$$
3. Compute the following:
$$\mathsf{ct}_0^* = \mathsf{FHE.Eval}(\mathsf{pk}_0, \mathsf{ct}_{1,0}, \mathsf{ct}_{2,0}, \mathsf{C}_{\mathsf{Add,FHE}}) \quad , \quad \mathsf{ct}_1^* = \mathsf{FHE.Eval}(\mathsf{pk}_1, \mathsf{ct}_{1,1}, \mathsf{ct}_{2,1}, \mathsf{C}_{\mathsf{Add,FHE}}).$$
4. **If** any of the following is true:
   (a) $a_{0,0,1,0} \neq a_{0,0,1,1}$ or $a_{0,0,2,0} \neq a_{0,0,2,1}$.
   (b) There exists some $(\ell, \ell') \neq (0,0)$ and some $(t,b) \in \{1,2\} \times \{0,1\}$ such that $a_{\ell,\ell',t,b} \neq 0$.
   (c) There exists some $(t,b) \in \{1,2\} \times \{0,1\}$ such that $i_{t,b} \neq i$.
   then proceed as follows:
   (a) Extract $\mathsf{wit}_y = \mathsf{NIZK.Ext}(\mathsf{t}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_{1,0}, \mathsf{ct}_{1,1}, i, y), \pi_1)$.
   (b) **If** $\mathsf{R}_{\mathcal{L}}(\mathsf{wit}_y, y) = 0$, output $\perp$.
   (c) **Else**, generate $\pi^* \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0^*, \mathsf{ct}_1^*, i, y), \mathsf{wit}_y)$.
5. **Else**, generate $\pi^* \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0^*, \mathsf{ct}_1^*, i, y), (\mathsf{sk}_0, \mathsf{sk}_1))$.
6. Output $(\mathsf{ct}_0^*, \mathsf{ct}_1^*, i, \pi^*)$.

Figure 27: Let $\mathsf{crs} \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$. We now assume that crs is in binding mode. Let $\mathsf{NIZK.Ext}$ be the extraction algorithm for NIZK. Let t be the extraction trapdoor corresponding to crs. Additionally, let $(\mathsf{pk}_0, \mathsf{sk}_0), (\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathsf{FHE.Gen}(1^\lambda)$. We define a probabilistic circuit $\mathsf{C}_{\mathsf{Add,MMap}}$ (which has t and the key pairs $(\mathsf{pk}_0, \mathsf{sk}_0)$ and $(\mathsf{pk}_1, \mathsf{sk}_1)$ hard-wired into it) as above.

Inputs: A tuple of the form $(a, \{a_{\ell,\ell'}\}_{(\ell+\ell')\in[0,N]}, i)$.

Computation: If $i > N$, output $\perp$ else output $(\{-a_{\ell,\ell'}\}_{(\ell+\ell')\in[0,N]}, i)$.

Figure 28: Circuit $\mathsf{C}_{\mathsf{Inv,FHE}}$

Conceptually, we multiply these two encodings by again exploiting the fully-homomorphic nature of the encryption scheme. More concretely, we homomorphically evaluate the circuit $\mathsf{C}_{\mathsf{Mult,FHE}}$ (described in Figure 30) on the corresponding ciphertext components of the two input encodings to generate the ciphertext components for the output encoding. We also generate a proof for the validity of the resulting encoding using the tuple of secret keys $(\mathsf{sk}_0, \mathsf{sk}_1)$ as witness.

Similar to the addition and inversion procedures, for technical reasons that are relevant to the proof of security, we check the validity and consistency of the input encodings. Figure 7 details the operation of the encoding-multiplication circuit $\mathsf{C}_{\mathsf{Mult,MMap}}$. Once again, it embeds multiple secrets, including the secret keys sk_0 and sk_1, as well as the extraction trapdoor t for the NIZK. Hence, the circuit is not made public as is; we only make available an obfuscated version of the circuit obtained by running the evaluation algorithm of the probabilistic iO scheme piO on it.

We emphasize that when the element $y \in \mathcal{X}$ is a non-member for the language $\mathcal{L}$, then under a binding crs, the "**If**"

Encoding Inversion Circuit: $\mathsf{C}_{\mathsf{Inv,MMap}}$

Inputs: A tuple of the form: $(\mathsf{ct}_0, \mathsf{ct}_1, i, \pi, y \in \mathcal{X})$, where the proof π is over the language L.

Outputs: Either a tuple of the form $(\mathsf{ct}_0^*, \mathsf{ct}_1^*, i, \pi^*)$, or $\perp$.

Computation: Proceed as follows:

1. If $\mathsf{NIZK.Verify}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0, \mathsf{ct}_1, i, y), \pi) = 0$, then output $\perp$.
2. Recover the following:
$$(\{a_{\ell,\ell',0}\}_{(\ell+\ell')\in[0,N]}, i_0) = \mathsf{FHE.Dec}(\mathsf{sk}_0, \mathsf{ct}_0) \quad , \quad (\{a_{\ell,\ell',1}\}_{(\ell+\ell')\in[0,N]}, i_1) = \mathsf{FHE.Dec}(\mathsf{sk}_1, \mathsf{ct}_1).$$
3. Compute the following:
$$\mathsf{ct}_0^* = \mathsf{FHE.Eval}(\mathsf{pk}_0, \mathsf{ct}_0, \mathsf{C}_{\mathsf{Inv,FHE}}) \quad , \quad \mathsf{ct}_1^* = \mathsf{FHE.Eval}(\mathsf{pk}_1, \mathsf{ct}_1, \mathsf{C}_{\mathsf{Inv,FHE}}).$$
4. If any of the following is true:
   (a) $a_{0,0,0} \neq a_{0,0,1}$.
   (b) There exists some $(\ell, \ell') \neq (0,0)$ and some $b \in \{0,1\}$ such that $a_{\ell,\ell',b} \neq 0$.
   (c) There exists some $b \in \{0,1\}$ such that $i_b \neq i$.

   then proceed as follows:
   (a) Extract $\mathsf{wit}_y = \mathsf{NIZK.Ext}(\mathsf{t}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0, \mathsf{ct}_1, i, y), \pi)$.
   (b) **If** $\mathsf{R}_{\mathcal{L}}(\mathsf{wit}_y, y) = 0$, output $\perp$.
   (c) **Else**, generate $\pi^* \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0^*, \mathsf{ct}_1^*, i, y), \mathsf{wit}_y)$.
5. **Else**, generate $\pi^* \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0^*, \mathsf{ct}_1^*, i, y), (\mathsf{sk}_0, \mathsf{sk}_1))$.
6. Output $(\mathsf{ct}_0^*, \mathsf{ct}_1^*, i, \pi^*)$.

Figure 29: Let $\mathsf{crs} \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$. We now assume that crs is in binding mode. Let $\mathsf{NIZK.Ext}$ be the extraction algorithm for NIZK. Let t be the extraction trapdoor corresponding to crs. Additionally, let $(\mathsf{pk}_0, \mathsf{sk}_0), (\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathsf{FHE.Gen}(1^\lambda)$. We define a probabilistic circuit $\mathsf{C}_{\mathsf{Inv,MMap}}$ (which has t and the key pairs $(\mathsf{pk}_0, \mathsf{sk}_0)$ and $(\mathsf{pk}_1, \mathsf{sk}_1)$ hard-wired into it) as above.

Inputs: A tuple of the form $((\{a_{\ell,\ell',1}\}_{(\ell+\ell')\in[0,N]}, i_1), (\{a_{\ell,\ell',2}\}_{(\ell+\ell')\in[0,N]}, i_2))$.

Computation: If $i_1 + i_2 > N$, output $\perp$. Else, output $(\{a^*_{\ell,\ell'}\}_{(\ell+\ell')\in[0,N]}, (i_1 + i_2))$, where
$$a^*_{\ell,\ell'} = \sum_{\substack{\ell_1+\ell_2=\ell \\ \ell'_1+\ell'_2=\ell'}} a_{\ell_1,\ell'_1,1} \cdot a_{\ell_2,\ell'_2,2}.$$

Figure 30: Circuit $\mathsf{C}_{\mathsf{Mult,FHE}}$

condition in step 4 of $\mathsf{C}_{\mathsf{Mult,MMap}}$ is never satisfied. This follows from the perfect extractability guarantee of the NIZK proof system. However, the condition may be satisfied during some hybrid in the proof of security, under a hiding crs when the element $y \in \mathcal{X}$ is a member for the language $\mathcal{L}$.

This in turn guarantees that when the element $y \in \mathcal{X}$ is a non-member for the language $\mathcal{L}$, then under a binding crs, no encoding (even adversarially generated) can have "auxiliary" FHE ciphertexts encrypting other than the plaintext element 0. Hence, during multiplication, we do not need to worry about computing and adding "cross-product" terms across the input ciphertexts.

6.5 Extraction and Zero-Testing

Extraction. We now describe the procedure for extracting a canonical string from an encoding at any given level-vector. Suppose we have an encodings at the level i of the form: $(\mathsf{ct}_0, \mathsf{ct}_1, i, \pi)$. The straightforward way of extracting a canonical representation is to decrypt the corresponding ciphertexts using sk_0 and sk_1, and outputting $g^* = g^{a_{0,0,0}} =$

Encoding Multiplication Circuit: $\mathsf{C}_{\mathsf{Mult},\mathbf{MMap}}$

Inputs: A tuple of the form: $((\mathsf{ct}_{1,0}, \mathsf{ct}_{1,1}, i_1, \pi_1), (\mathsf{ct}_{2,0}, \mathsf{ct}_{2,1}, i_2, \pi_2), y \in \mathcal{X})$, where the proofs π_1 and π_2 are over the language L.

Outputs: Either a tuple of the form $(\mathsf{ct}_0^*, \mathsf{ct}_1^*, (i_1 + i_2), \pi^*)$, or $\perp$.

Computation: Proceed as follows:

1. **If** any of the following is true:
   (a) $\mathsf{NIZK.Verify}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_{1,0}, \mathsf{ct}_{1,1}, i_1, y \in \mathcal{X}), \pi_1) = 0$.
   (b) $\mathsf{NIZK.Verify}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_{2,0}, \mathsf{ct}_{2,1}, i_2, y \in \mathcal{X}), \pi_2) = 0$.

   then output $\perp$.
2. Recover the following:

$$(\{a_{\ell,\ell',1,0}\}_{(\ell+\ell')\in[0,N]}, i_{1,0}) = \mathsf{FHE.Dec}(\mathsf{sk}_0, \mathsf{ct}_{1,0}), (\{a_{\ell,\ell',1,1}\}_{(\ell+\ell')\in[0,N]}, i_{1,1}) = \mathsf{FHE.Dec}(\mathsf{sk}_1, \mathsf{ct}_{1,1}),$$

$$(\{a_{\ell,\ell',2,0}\}_{(\ell+\ell')\in[0,N]}, i_{2,0}) = \mathsf{FHE.Dec}(\mathsf{sk}_0, \mathsf{ct}_{2,0}), (\{a_{\ell,\ell',2,1}\}_{(\ell+\ell')\in[0,N]}, i_{2,1}) = \mathsf{FHE.Dec}(\mathsf{sk}_1, \mathsf{ct}_{2,1}).$$

3. Compute the following:

$$\mathsf{ct}_0^* = \mathsf{FHE.Eval}(\mathsf{pk}_0, \mathsf{ct}_{1,0}, \mathsf{ct}_{2,0}, \mathsf{C}_{\mathsf{Mult,FHE}}) \quad , \quad \mathsf{ct}_1^* = \mathsf{FHE.Eval}(\mathsf{pk}_1, \mathsf{ct}_{1,1}, \mathsf{ct}_{2,1}, \mathsf{C}_{\mathsf{Mult,FHE}}).$$

4. **If** any of the following is true:
   (a) $a_{0,0,1,0} \neq a_{0,0,1,1}$ or $a_{0,0,2,0} \neq a_{0,0,2,1}$.
   (b) There exists some $(\ell, \ell') \neq (0,0)$ and some $(t, b) \in \{1,2\} \times \{0,1\}$ such that $a_{\ell,\ell',t,b} \neq 0$.
   (c) There exists some $b \in \{0,1\}$ such that $i_{1,b} \neq i_1$ or $i_{2,b} \neq i_2$.

   then proceed as follows:
   (a) Extract $\mathsf{wit}_y = \mathsf{NIZK.Ext}(\mathsf{t}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_{1,0}, \mathsf{ct}_{1,1}, i, y), \pi_1)$.
   (b) **If** $\mathsf{R}_{\mathcal{L}}(\mathsf{wit}_y, y) = 0$, output $\perp$.
   (c) **Else**, generate $\pi^* \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0^*, \mathsf{ct}_1^*, (i_1 + i_2), y), \mathsf{wit}_y)$.
5. **Else**, generate $\pi^* \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0^*, \mathsf{ct}_1^*, (i_1 + i_2), y), (\mathsf{sk}_0, \mathsf{sk}_1))$.
6. Output $(\mathsf{ct}_0^*, \mathsf{ct}_1^*, (i_1 + i_2), \pi^*)$.

Figure 31: Let $\mathsf{crs} \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$. We now assume that crs is in binding mode. Let $\mathsf{NIZK.Ext}$ be the extraction algorithm for NIZK. Let t be the extraction trapdoor corresponding to crs. Additionally, let $(\mathsf{pk}_0, \mathsf{sk}_0), (\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathsf{FHE.Gen}(1^\lambda)$. We define a probabilistic circuit $\mathsf{C}_{\mathsf{Mult,MMap}}$ (which has t and the key pairs $(\mathsf{pk}_0, \mathsf{sk}_0)$ and $(\mathsf{pk}_1, \mathsf{sk}_1)$ hard-wired into it) as in above.

$g^{a_{0,0,1}}$, where g is a fixed group element embedded inside the extraction circuit. However, for technical reasons relevant to the proof of security, we sightly depart from this straightforward process.

More concretely, our extraction circuit embeds a tuple of the form $(\{g_{\ell,\ell'}\}_{(\ell+\ell')\in[0,N]})$, where each element of the tuple is a group element in the group $\mathbb{G}$. The circuit then computes

$$(\{a_{\ell,\ell',0}\}_{(\ell+\ell')\in[0,N]}, i_0) = \mathsf{FHE.Dec}(\mathsf{sk}_0, \mathsf{ct}_0),$$

and outputs a group element g^*, where

$$g^* = \prod_{(\ell+\ell')\in[0,N]} (g_{\ell,\ell'})^{a_{\ell,\ell',0}}.$$

It is easy to see that when the "auxiliary" terms in the underlying plaintext are all 0, this is functionally equivalent to the simpler extraction strategy described above. However, if one or more of the "auxiliary" plaintext terms is non-zero, the two processes are no longer equivalent. As it turns out, we exploit this fact in the proof of security.

Again, for technical reasons that are relevant to the proof of security, we also use the secret keys sk_0 and sk_1, as well as the NIZK extraction trapdoor t, to check: (a) the validity of the proofs π_1 and π_2 for the language L that are provided as part of the input encodings, and (b) the "consistency" of the FHE ciphertexts that are provided as part of the input encodings, i.e., whether the "auxiliary" set of plaintext elements are all zero and whether both ciphertexts encrypt the same level.

Extraction Circuit: $\mathsf{C}_{\mathsf{Ext,MMap}}$

Inputs: A tuple of the form: $\Big((\mathsf{ct}_0, \mathsf{ct}_1, i, \pi), y \in \mathcal{X}\Big)$, where the proof π is over the language L.

Outputs: Either a group element g^* or $\perp$.

Computation: Proceed as follows:

1. If $\mathsf{NIZK.Verify}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0, \mathsf{ct}_1, i, y), \pi) = 0$, then output $\perp$.
2. Recover the following:

$$(\{a_{\ell,\ell',0}\}_{(\ell+\ell')\in[0,N]}, i_0) = \mathsf{FHE.Dec}(\mathsf{sk}_0, \mathsf{ct}_0) \quad , \quad (\{a_{\ell,\ell',1}\}_{(\ell+\ell')\in[0,N]}, i_1) = \mathsf{FHE.Dec}(\mathsf{sk}_1, \mathsf{ct}_1).$$

3. Compute $g^* = \prod_{(\ell+\ell')\in[0,N]} (g_{\ell,\ell'})^{a_{\ell,\ell',0}}$.
4. If any of the following is true:
   (a) $a_{0,0,0} \neq a_{0,0,1}$.
   (b) There exists some $(\ell, \ell') \neq (0,0)$ and some $b \in \{0,1\}$ such that $a_{\ell,\ell',b} \neq 0$.
   (c) There exists some $b \in \{0,1\}$ such that $i_b \neq i$.

   then proceed as follows:
   (a) Extract $\mathsf{wit}_y = \mathsf{NIZK.Ext}(\mathsf{t}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0, \mathsf{ct}_1, i, y), \pi)$.
   (b) **If** $\mathsf{R}_{\mathcal{L}}(\mathsf{wit}_y, y) = 0$, output $\perp$.
5. **Else**, output g^*.

Figure 32: Let $\mathsf{crs} \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$. We now assume that crs is in binding mode. Let $\mathsf{NIZK.Ext}$ be the extraction algorithm for NIZK. Let t be the extraction trapdoor corresponding to crs. Additionally, let $(\mathsf{pk}_0, \mathsf{sk}_0), (\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathsf{FHE.Gen}(1^\lambda)$, and let $\{g_{\ell,\ell'}\}_{(\ell+\ell')\in[0,N]}$ be a set of arbitrary group elements over the group $\mathbb{G}$. We define a deterministic circuit $\mathsf{C}_{\mathsf{Ext,MMap}}$ (which has t, the key pairs $(\mathsf{pk}_0, \mathsf{sk}_0)$ and $(\mathsf{pk}_1, \mathsf{sk}_1)$, and the set of group elements $\{g_{\ell,\ell'}\}_{(\ell+\ell')\in[0,N]}$ hard-wired into it) as above.

Figure 32 details the operation of the extraction circuit $\mathsf{C}_{\mathsf{Ext,MMap}}$. As before, the circuit $\mathsf{C}_{\mathsf{Ext,MMap}}$ is not made public as is; we only make available an obfuscated version of the circuit obtained by running the evaluation algorithm of the probabilistic iO scheme piO on it.

Finally, we emphasize that when the element $y \in \mathcal{X}$ is a non-member for the language $\mathcal{L}$, then under a binding crs, the "**If**" condition in step 4 of $\mathsf{C}_{\mathsf{Ext,MMap}}$ is never satisfied. This follows from the perfect extractability guarantee of the NIZK proof system. However, the condition may be satisfied during some hybrid in the proof of security, under a hiding crs when the element $y \in \mathcal{X}$ is a member for the language $\mathcal{L}$.

Zero-Testing Encodings. Given the aforementioned extraction procedure, zero-testing an encoding at any given level is trivial. We simply apply the extraction procedure to the encoding, and check if the extracted group element g^* is equal to g^0 for any $g \in \mathbb{G}$.

6.6 The Overall Construction

We are now ready to formally summarize our construction of an $(N+1)$-EDDH-hard symmetric MMap.

- $\mathsf{Setup}(1^\lambda, N)$: On input the security parameter λ and the degree of multilinearity N, do the following:
  1. Sample (in binding mode) $(\mathsf{crs}, \mathsf{t}) \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$.
  2. Sample $(\mathsf{pk}_0, \mathsf{sk}_0), (\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathsf{FHE.Gen}(1^\lambda)$.
  3. Sample a non-member instance $y \leftarrow \mathcal{X} \setminus \mathcal{L}$.
  4. Compute $\mathsf{C}_{\mathsf{Add,piO}} = \mathsf{piO.Obf}(\mathsf{C}_{\mathsf{Add,MMap}})$, where $\mathsf{C}_{\mathsf{Add,MMap}}$ is as defined in Figure 3.
  5. Compute $\mathsf{C}_{\mathsf{Inv,piO}} = \mathsf{piO.Obf}(\mathsf{C}_{\mathsf{Inv,MMap}})$, where $\mathsf{C}_{\mathsf{Inv,MMap}}$ is as defined in Figure 5.

6. Compute $\mathsf{C}_{\mathsf{Mult,piO}} = \mathsf{piO.Obf}(\mathsf{C}_{\mathsf{Mult,MMap}})$, where $\mathsf{C}_{\mathsf{Mult,MMap}}$ is as defined in Figure 7.
7. Compute $\mathsf{C}_{\mathsf{Ext,piO}} = \mathsf{piO.Obf}(\mathsf{C}_{\mathsf{Ext,MMap}})$, where $\mathsf{C}_{\mathsf{Ext,MMap}}$ is as defined in Figure 8.
8. Output the public parameter pp, where

$$\mathsf{pp} = \left(\mathsf{crs}, \mathsf{pk}_0, \mathsf{pk}_1, \mathsf{C}_{\mathsf{Add,piO}}, \mathsf{C}_{\mathsf{Inv,piO}}, \mathsf{C}_{\mathsf{Mult,piO}}, \mathsf{C}_{\mathsf{Ext,piO}}, \{g_{\ell,\ell'}\}_{(\ell+\ell')\in[0,N]}, y\right).$$

- $\mathsf{Encode}(\mathsf{pp}, a, i)$: On input the public parameter pp, a plaintext element $a \in \mathbb{Z}_q$ and $i \in [N]$, proceed as follows:
  1. Set $a_{0,0} = a$.
  2. For each $(\ell, \ell') \neq (0,0)$ such that $(\ell + \ell') \in [0, N]$, set $a_{\ell,\ell'} = 0$.
  3. Generate the following ciphertexts:

$$\mathsf{ct}_0 = \mathsf{FHE.Enc}(\mathsf{pk}_0, \{a_{\ell,\ell'}\}_{(\ell+\ell')\in[0,N]}; \mathsf{r}_0) \quad , \quad \mathsf{ct}_1 \leftarrow \mathsf{FHE.Enc}(\mathsf{pk}_1, \{a_{\ell,\ell'}\}_{(\ell+\ell')\in[0,N]}; \mathsf{r}_1),$$

     where r_0 and r_1 denote uniformly sampled random coins.
  4. Generate the following proof:

$$\pi \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, \mathsf{st}, \mathsf{wit}),$$

     where the statement st and the witness wit correspond to the NP language L (as defined in Figure 25) and are given as:

$$\mathsf{st} = (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0, \mathsf{ct}_1, i, y) \quad , \quad \mathsf{wit} = (a, \mathsf{r}_0, \mathsf{r}_1).$$

  5. Finally, output the encoding

$$[a]_i := (\mathsf{ct}_0, \mathsf{ct}_1, i, \pi).$$

- $\mathsf{Add}(\mathsf{pp}, [a_1]_i, [a_2]_i)$: On input the public parameter pp and a pair of encodings

$$[a_1]_i = (\mathsf{ct}_{1,0}, \mathsf{ct}_{1,1}, i, \pi_1) \quad , \quad [a_2]_i = (\mathsf{ct}_{2,0}, \mathsf{ct}_{2,1}, i, \pi_2),$$

  output the encoding $[a_1 + a_2]_i = (\mathsf{ct}^*_0, \mathsf{ct}^*_1, i, \pi^*)$, where

$$(\mathsf{ct}^*_0, \mathsf{ct}^*_1, i, \pi^*) = \mathsf{piOEval}(\mathsf{C}_{\mathsf{Add,MMap}}, ((\mathsf{ct}_{1,0}, \mathsf{ct}_{1,1}, i, \pi_1), (\mathsf{ct}_{2,0}, \mathsf{ct}_{2,1}, i, \pi_2), y)).$$

- $\mathsf{Inv}(\mathsf{pp}, [a]_i)$: On input the public parameter pp and an encoding $[[a]_i = (\mathsf{ct}_0, \mathsf{ct}_1, i, \pi)$, output the encoding $[-a]_i = (\mathsf{ct}^*_0, \mathsf{ct}^*_1, i, \pi^*)$, where

$$(\mathsf{ct}^*_0, \mathsf{ct}^*_1, i, \pi^*) = \mathsf{piOEval}(\mathsf{C}_{\mathsf{Inv,MMap}}, (\mathsf{ct}_0, \mathsf{ct}_1, i, \pi, y)).$$

- $\mathsf{Mult}(\mathsf{pp}, [a_1]_i, [a_2]_j)$: On input the public parameter pp and a pair of encodings

$$[a_1]_{i_1} = (\mathsf{ct}_{1,0}, \mathsf{ct}_{1,1}, i, \pi_1), \quad , \quad [a_2]_{i_2} = (\mathsf{ct}_{2,0}, \mathsf{ct}_{2,1}, j+1, k, \pi_2),$$

  such that $(i_1 + i_2) \leq N$, output the encoding $[a_1.a_2]_{i_1+i_2} = (\mathsf{ct}^*_0, \mathsf{ct}^*_1, (i_1 + i_2), \pi^*)$, where

$$(\mathsf{ct}^*_0, \mathsf{ct}^*_1, (i_1 + i_2), \pi^*) = \mathsf{piOEval}(\mathsf{C}_{\mathsf{Mult,MMap}}, ((\mathsf{ct}_{1,0}, \mathsf{ct}_{1,1}, i, \pi_1), (\mathsf{ct}_{2,0}, \mathsf{ct}_{2,1}, j+1, k, \pi_2), y)).$$

- $\mathsf{Ext}(\mathsf{pp}, [a]_i)$: On input the public parameter pp and an encoding $[a]_i = (\mathsf{ct}_0, \mathsf{ct}_1, i, \pi)$, output $b \in \{0, 1, \perp\}$, where

$$b = \mathsf{piOEval}\left(\mathsf{C}_{\mathsf{Ext,MMap}}, ((\mathsf{ct}_0, \mathsf{ct}_1, i, \pi), \{g_{\ell,\ell'}\}_{(\ell+\ell')\in[0,N]}, y)\right).$$

6.7 Proof of $(N+1)$-EDDH

Recall that the $(N+1)$-EDDH assumption over a degree-N symmetric MMap requires that the following distribution ensembles are computationally indistinguishable from random for any level $i \in [N]$:

$$\left([\alpha]_i, [\alpha^{N+1}]_i\right) \quad \approx_c \quad \left([\alpha]_i, [\alpha^*]_i\right),$$

where $\alpha, \alpha^* \leftarrow \mathbb{Z}_q$. For simplicity, we prove here the version of the assumption where α and α^* are sampled uniformly from $\mathbb{Z}_q^*$ (which is statistically indistinguishable from the case where α and α^* are sampled uniformly from $\mathbb{Z}_q$ whenever $q = 2^{\omega(\log \lambda)}$).

We state and prove the following theorem:

Theorem 6.1. *Our symmetric MMap construction for degree-N is $(N+1)$-EDDH-hard assuming: (a) that deciding subset membership with respect to $\mathcal{L} \subset \mathcal{X}$ is computationally hard, (b) that NIZK satisfies* crs*-indistinguishability, perfect simulation-extractability under a binding* crs *and perfect zero-knowledge under a hiding* crs*, (c) that piO satisfies probabilistic-iO security, (d) that FHE is IND-CPA secure, and (e) that the group $\mathbb{G}$ is "strong" $(N+1)$-EDDH-hard.*

The proof again proceeds via a sequence of hybrids summarized in Table 4. As was the case for our asymmetric MMap constructions, the crux of the proof lies in switching the plaintexts underlying the challenge encodings in the $(N+1)$-EDDH game to a special form such that:

- The challenge encodings constitute a valid $(N+1)$-EDDH instance over the MMap when the extraction circuit embeds a valid "strong" $(N+1)$-EDDH tuple over the group $\mathbb{G}$.
- The challenge encodings constitute a random invalid $(N+1)$-EDDH instance over the MMap when the extraction circuit embeds a random invalid "strong" $(N+1)$-EDDH tuple over the group $\mathbb{G}$.

We then show that these two cases are indistinguishable based on a sequence of hybrids, where each pair of consecutive hybrids is indistinguishable based on the computational security properties of the underlying cryptoprimitives and the hardness of strong $(N+1)$-EDDH over the group $\mathbb{G}$.

Hybrid-0. In this hybrid, the public parameter pp is generated as in the real MMap scheme described above. As part of the $(N+1)$-EDDH challenge, the challenger provides the adversary with encodings of α (uniformly random in $\mathbb{Z}_q^*$) and α^{N+1}, generated as per the real encoding algorithm corresponding to challenge interval i (may be adversarially chosen).

Hybrid-1. In this hybrid, we change the public parameter pp provided to the adversary as follows:

1. Switch crs for the NIZK proof system to hiding mode from binding mode.
2. Switch $y \in \mathcal{X}$ from a uniform non-member instance to a uniform member instance for the language $\mathcal{L}$ with witness wit_y.
3. Compute $\mathsf{C}_{\mathsf{Add},\mathsf{piO}} = \mathsf{piO.Obf}(\widehat{\mathsf{C}}_{\mathsf{Add},\mathsf{MMap}})$, where $\widehat{\mathsf{C}}_{\mathsf{Add},\mathsf{MMap}}$ is as described in Figure 33.
4. Compute $\mathsf{C}_{\mathsf{Inv},\mathsf{piO}} = \mathsf{piO.Obf}(\widehat{\mathsf{C}}_{\mathsf{Inv},\mathsf{MMap}})$, where $\widehat{\mathsf{C}}_{\mathsf{Inv},\mathsf{MMap}}$ is as described in Figure 34.
5. Compute $\mathsf{C}_{\mathsf{Mult},\mathsf{piO}} = \mathsf{piO.Obf}(\widehat{\mathsf{C}}_{\mathsf{Mult},\mathsf{MMap}})$, where $\widehat{\mathsf{C}}_{\mathsf{Mult},\mathsf{MMap}}$ is as described in Figure 35.
6. Compute $\mathsf{C}_{\mathsf{Ext},\mathsf{piO}} = \mathsf{piO.Obf}(\widehat{\mathsf{C}}_{\mathsf{Ext},\mathsf{MMap},1})$, where $\widehat{\mathsf{C}}_{\mathsf{Ext},\mathsf{MMap},1}$ is as described in Figure 36.

Lemma 6.2. *Hybrid-1 is computationally indistinguishable from Hybrid-0 assuming: (a) that deciding subset membership with respect to $\mathcal{L} \subset \mathcal{X}$ is computationally hard, (b) that NIZK satisfies* crs*-indistinguishability, perfect simulation-extractability under a binding* crs *and perfect zero-knowledge under a hiding* crs*, and (c) that piO satisfies probabilistic-iO security.*

Table 4: Overview of hybrids for proof of $(N+1)$-EDDH

Hybrid	crs	y	$\mathsf{C}_{\phi,\mathsf{MMap}}$ knows for $\phi \in \{\mathsf{Add}, \mathsf{Inv}, \mathsf{Mult}\}$	$\mathsf{C}_{\mathsf{Ext},\mathsf{MMap}}$ knows	$\{g_{\ell,\ell'}\}$ is of the form	Challenge FHE Ciphertexts (Set-1) are encryptions of	Challenge FHE Ciphertexts (Set-2) are encryptions of	Comments
0	binding	$\notin \mathcal{L}$	$(\mathsf{sk}_0, \mathsf{sk}_1, \mathsf{t})$	$(\mathsf{sk}_0, \mathsf{sk}_1, \mathsf{t})$	$\left\{g^{\gamma^\ell \cdot \delta^{\ell'}}\right\}$	$(\alpha, 0, 0, 0, \ldots, 0)$ $(\alpha^{N+1}, 0, 0, 0, \ldots, 0)$	$(\alpha, 0, 0, 0, \ldots, 0)$ $(\alpha^{N+1}, 0, 0, 0, \ldots, 0)$	—
1	hiding	$\in \mathcal{L}$	—	sk_1	$\left\{g^{\gamma^{\ell+(N+1)\ell'}}\right\}$	$(\alpha, 0, 0, 0, \ldots, 0)$ $(\alpha^{N+1}, 0, 0, 0, \ldots, 0)$	$(\alpha, 0, 0, 0, \ldots, 0)$ $(\alpha^{N+1}, 0, 0, 0, \ldots, 0)$	NIZK security + subspace-membership hardness + piO security
2	hiding	$\in \mathcal{L}$	—	sk_1	$\left\{g^{\gamma^{\ell+(N+1)\ell'}}\right\}$	$(0, \alpha \cdot \gamma^{-1}, 0, 0, \ldots, 0)$ $(0, 0, (\alpha \cdot \gamma^{-1})^{N+1}, 0, \ldots, 0)$	$(\alpha, 0, 0, 0, \ldots, 0)$ $(\alpha^{N+1}, 0, 0, 0, \ldots, 0)$	FHE IND-CPA security
3	hiding	$\in \mathcal{L}$	—	sk_0	$\left\{g^{\gamma^{\ell+(N+1)\ell'}}\right\}$	$(0, \alpha \cdot \gamma^{-1}, 0, 0, \ldots, 0)$ $(0, 0, (\alpha \cdot \gamma^{-1})^{N+1}, 0, \ldots, 0)$	$(\alpha, 0, 0, 0, \ldots, 0)$ $(\alpha^{N+1}, 0, 0, 0, \ldots, 0)$	NIZK security + piO security
4	hiding	$\in \mathcal{L}$	—	sk_0	$\left\{g^{\gamma^{\ell+(N+1)\ell'}}\right\}$	$(0, \alpha \cdot \gamma^{-1}, 0, 0, \ldots, 0)$ $(0, 0, (\alpha \cdot \gamma^{-1})^{N+1}, 0, \ldots, 0)$	$(0, \alpha \cdot \gamma^{-1}, 0, 0, \ldots, 0)$ $(0, 0, (\alpha \cdot \gamma^{-1})^{N+1}, 0, \ldots, 0)$	FHE IND-CPA security
5	hiding	$\in \mathcal{L}$	—	sk_0	$\left\{g^{\gamma^\ell \cdot \delta^{\ell'}}\right\}$	$(0, \alpha \cdot \gamma^{-1}, 0, 0, \ldots, 0)$ $(0, 0, (\alpha \cdot \gamma^{-1})^{N+1}, 0, \ldots, 0)$	$(0, \alpha \cdot \gamma^{-1}, 0, 0, \ldots, 0)$ $(0, 0, (\alpha \cdot \gamma^{-1})^{N+1}, 0, \ldots, 0)$	strong $(N+1)$-EDDH
6	hiding	$\in \mathcal{L}$	—	sk_0	$\left\{g^{\gamma^\ell \cdot \delta^{\ell'}}\right\}$	$(0, \alpha \cdot \gamma^{-1}, 0, 0, \ldots, 0)$ $(0, 0, (\alpha \cdot \gamma^{-1})^{N+1}, 0, \ldots, 0)$	$(\alpha, 0, 0, 0, \ldots, 0)$ $((\alpha \cdot \gamma^{-1})^{N+1} \cdot \delta, 0, 0, 0, \ldots, 0)$	FHE IND-CPA security
7	hiding	$\in \mathcal{L}$	—	sk_1	$\left\{g^{\gamma^\ell \cdot \delta^{\ell'}}\right\}$	$(0, \alpha \cdot \gamma^{-1}, 0, 0, \ldots, 0)$ $(0, 0, (\alpha \cdot \gamma^{-1})^{N+1}, 0, \ldots, 0)$	$(\alpha, 0, 0, 0, \ldots, 0)$ $((\alpha \cdot \gamma^{-1})^{N+1} \cdot \delta, 0, 0, 0, \ldots, 0)$	NIZK security + piO security
8	hiding	$\in \mathcal{L}$	—	sk_1	$\left\{g^{\gamma^\ell \cdot \delta^{\ell'}}\right\}$	$(\alpha, 0, 0, 0, \ldots, 0)$ $((\alpha \cdot \gamma^{-1})^{N+1} \cdot \delta, 0, 0, 0, \ldots, 0)$	$(\alpha, 0, 0, 0, \ldots, 0)$ $((\alpha \cdot \gamma^{-1})^{N+1} \cdot \delta, 0, 0, 0, \ldots, 0)$	FHE IND-CPA security
9	binding	$\notin \mathcal{L}$	$\left\{g^{\gamma^\ell \cdot \delta^{\ell'}}\right\}$	$(\mathsf{sk}_0, \mathsf{sk}_1, \mathsf{t})$	$(\mathsf{sk}_0, \mathsf{sk}_1, \mathsf{t})$	$(\alpha, 0, 0, 0, \ldots, 0)$ $((\alpha \cdot \gamma^{-1})^{N+1} \cdot \delta, 0, 0, 0, \ldots, 0)$	$(\alpha, 0, 0, 0, \ldots, 0)$ $((\alpha \cdot \gamma^{-1})^{N+1} \cdot \delta, 0, 0, 0, \ldots, 0)$	NIZK security + piO security

The proof of this lemma is very similar to the proof of Lemma 4.2 and is hence not detailed.

Note 6.3. In the remaining hybrid descriptions, we assume (for ease of representation) that the plaintext underlying each FHE ciphertext in an encoding is arranged as $(a_{0,0}, a_{1,0}, a_{0,1}, a_{2,0}, a_{1,1}, a_{0,2}, \ldots, a_{0,N}, i)$.

Hybrid-2. In this hybrid, we change the manner in which the $(N+1)$-EDDH challenge encodings are generated. Suppose that the set of group elements $\{g_{\ell,\ell'}\}_{(\ell+\ell')\in[0,N]}$ embedded in the extraction circuit $\widehat{C}_{\mathsf{Ext},\mathsf{MMap},1,0}$ is distributed as:

$$\{g_{\ell,\ell'}\}_{(\ell+\ell')\in[0,N]} = \left\{g^{\gamma^{\ell+(N+1)\ell'}}\right\}_{(\ell+\ell')\in[0,N]}.$$

We switch the plaintext underlying the *first* FHE ciphertext in each challenge encoding as follows:

$$\begin{aligned}(\alpha, 0, 0, 0, \ldots, 0, i) &\rightarrow (0, (\alpha \cdot \gamma^{-1}), 0, 0, \ldots, 0, i)\\ (\alpha^{N+1}, 0, 0, 0, \ldots, 0, i) &\rightarrow (0, 0, (\alpha \cdot \gamma^{-1})^{N+1}, 0, \ldots, 0, i)\end{aligned}$$

Inputs: A tuple of the form: $((\mathsf{ct}_{1,0}, \mathsf{ct}_{1,1}, i, \pi_1), (\mathsf{ct}_{2,0}, \mathsf{ct}_{2,1}, i, \pi_2), y \in \mathcal{X})$, where the proofs π_1 and π_2 are over the language L.

Outputs: Either a tuple of the form $(\mathsf{ct}_0^*, \mathsf{ct}_1^*, i, \pi^*)$, or $\perp$.

Computation: Proceed as follows:

1. **If** any of the following is true:
   (a) $\mathsf{NIZK.Verify}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_{1,0}, \mathsf{ct}_{1,1}, i, y \in \mathcal{X}), \pi_1) = 0$.
   (b) $\mathsf{NIZK.Verify}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_{2,0}, \mathsf{ct}_{2,1}, i, y \in \mathcal{X}), \pi_2) = 0$.
   then output $\perp$.
2. // Decryption omitted — avoids dependency on sk_0 and sk_1.
3. Compute the following:
$$\mathsf{ct}_0^* = \mathsf{FHE.Eval}(\mathsf{pk}_0, \mathsf{ct}_{1,0}, \mathsf{ct}_{2,0}, \mathsf{C}_{\mathsf{Add,FHE}}) \quad , \quad \mathsf{ct}_1^* = \mathsf{FHE.Eval}(\mathsf{pk}_1, \mathsf{ct}_{1,1}, \mathsf{ct}_{2,1}, \mathsf{C}_{\mathsf{Add,FHE}}).$$
4. // If check omitted — avoids dependency on sk_0 and sk_1.
   Proceed as follows:
   (a) // Extraction omitted — avoids dependency on extraction trapdoor t.
   (b) // Check omitted.
   (c) Generate $\pi^* \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0^*, \mathsf{ct}_1^*, i, y), \mathsf{wit}_y)$.
5. // Proof generation omitted — avoids dependency on sk_0 and sk_1.
6. Output $(\mathsf{ct}_0^*, \mathsf{ct}_1^*, i, \pi^*)$.

Figure 33: Let $\mathsf{crs} \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$ be in hiding mode. Additionally, let $(\mathsf{pk}_0, \mathsf{sk}_0), (\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathsf{FHE.Gen}(1^\lambda)$. Also, for $y \in \mathcal{L}$, let wit_y be a membership-witness such that $(y, \mathsf{wit}_y) \in \mathsf{R}_{\mathcal{L}}$. We define a probabilistic circuit $\widehat{\mathsf{C}}_{\mathsf{Add,MMap}}$ (which has only the public keys pk_0 and pk_1 hard-wired into it) as above.

A tuple of the form: $(\mathsf{ct}_0, \mathsf{ct}_1, i, \pi, y \in \mathcal{X})$, where the proof π is over the language L.

Outputs: Either a tuple of the form $(\mathsf{ct}_0^*, \mathsf{ct}_1^*, i, \pi^*)$, or $\perp$.

Computation: Proceed as follows:

1. If $\mathsf{NIZK.Verify}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0, \mathsf{ct}_1, i, y), \pi) = 0$, then outut $\perp$.
2. // Decryption omitted — avoids dependency on sk_0 and sk_1.
3. Compute the following:
$$\mathsf{ct}_0^* = \mathsf{FHE.Eval}(\mathsf{pk}_0, \mathsf{ct}_0, \mathsf{C}_{\mathsf{Inv,FHE}}) \quad , \quad \mathsf{ct}_1^* = \mathsf{FHE.Eval}(\mathsf{pk}_1, \mathsf{ct}_1, \mathsf{C}_{\mathsf{Inv,FHE}}).$$
4. // If check omitted — avoids dependency on sk_0 and sk_1.
   Proceed as follows:
   (a) // Extraction omitted — avoids dependency on extraction trapdoor t.
   (b) // Check omitted.
   (c) Generate $\pi^* \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0^*, \mathsf{ct}_1^*, i, y), \mathsf{wit}_y)$.
5. // Proof generation omitted — avoids dependency on sk_0 and sk_1.
6. Output $(\mathsf{ct}_0^*, \mathsf{ct}_1^*, i, \pi^*)$.

Figure 34: Next, Let $\mathsf{crs} \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$ be in hiding mode. Additionally, let $(\mathsf{pk}_0, \mathsf{sk}_0), (\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathsf{FHE.Gen}(1^\lambda)$. Also, for $y \in \mathcal{L}$, let wit_y be a membership-witness such that $(y, \mathsf{wit}_y) \in \mathsf{R}_{\mathcal{L}}$. We define a probabilistic circuit $\widehat{\mathsf{C}}_{\mathsf{Inv,MMap}}$ (which has only the public keys pk_0 and pk_1 hard-wired into it) as above

Lemma 6.4. *Hybrid-2 is computationally indistinguishable from Hybrid-1 assuming that FHE is IND-CPA secure with respect to the public key-secret key pair* $(\mathsf{pk}_0, \mathsf{sk}_0)$.

The proof of this lemma is very similar to the proof of Lemma 4.3 and is hence not detailed.

Hybrid-3. In this hybrid, we change the public parameter pp as follows: compute $\mathsf{C}_{\mathsf{Ext,piO}} = \mathsf{piO.Obf}(\widehat{\mathsf{C}}_{\mathsf{Ext,MMap},0,0})$, where $\widehat{\mathsf{C}}_{\mathsf{Ext,MMap},0,0}$ is as defined in Figure 36.

Inputs: A tuple of the form: $((\mathsf{ct}_{1,0}, \mathsf{ct}_{1,1}, i_1, \pi_1), (\mathsf{ct}_{2,0}, \mathsf{ct}_{2,1}, i_2, \pi_2), y \in \mathcal{X})$, where the proofs π_1 and π_2 are over the language L.

Outputs: Either a tuple of the form $(\mathsf{ct}^*_0, \mathsf{ct}^*_1, (i_1 + i_2), \pi^*)$, or $\perp$.

Computation: Proceed as follows:

1. **If** any of the following is true:
   (a) $\mathsf{NIZK.Verify}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_{1,0}, \mathsf{ct}_{1,1}, i, y \in \mathcal{X}), \pi_1) = 0$.
   (b) $\mathsf{NIZK.Verify}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_{2,0}, \mathsf{ct}_{2,1}, i, y \in \mathcal{X}), \pi_2) = 0$.
   then output $\perp$.
2. // Decryption omitted — avoids dependency on sk_0 and sk_1.
3. Compute the following:
$$\mathsf{ct}^*_0 = \mathsf{FHE.Eval}(\mathsf{pk}_0, \mathsf{ct}_{1,0}, \mathsf{ct}_{2,0}, \mathsf{C}_{\mathsf{Mult,FHE}}) \quad , \quad \mathsf{ct}^*_1 = \mathsf{FHE.Eval}(\mathsf{pk}_1, \mathsf{ct}_{1,1}, \mathsf{ct}_{2,1}, \mathsf{C}_{\mathsf{Mult,FHE}}).$$
4. // If check omitted — avoids dependency on sk_0 and sk_1.
   Proceed as follows:
   (a) // Extraction omitted — avoids dependency on extraction trapdoor t.
   (b) // Check omitted.
   (c) Generate $\pi^* \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}^*_0, \mathsf{ct}^*_1, i, k, y), \mathsf{wit}_y)$.
5. // Proof generation omitted — avoids dependency on sk_0 and sk_1.
6. Output $(\mathsf{ct}^*_0, \mathsf{ct}^*_1, i, k, \pi^*)$.

Figure 35: Let $\mathsf{crs} \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$ be in hiding mode. Additionally, let $(\mathsf{pk}_0, \mathsf{sk}_0), (\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathsf{FHE.Gen}(1^\lambda)$. Also, for $y \in \mathcal{L}$, let wit_y be a membership-witness such that $(y, \mathsf{wit}_y) \in \mathsf{R}_{\mathcal{L}}$. For each $b^* \in \{0, 1\}$, we define a probabilistic circuit $\widehat{\mathsf{C}}_{\mathsf{Mult,MMap}}$ (which has only the public keys pk_0 and pk_1 hard-wired into it) as above.

Inputs: A tuple of the form: $\Big((\mathsf{ct}_0, \mathsf{ct}_1, i, \pi), y \in \mathcal{X}\Big)$, where the proof π is over the language L.

Outputs: Either a group element g^* or $\perp$.

Computation: Proceed as follows:

1. If $\mathsf{NIZK.Verify}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0, \mathsf{ct}_1, i, y), \pi) = 0$, then output $\perp$.
2.
3. Recover the following: $(\{a_{\beta,\ell}\}_{\ell \in [0,3]}, i_\beta, j_\beta) = \mathsf{FHE.Dec}(\mathsf{sk}_\beta, \mathsf{ct}_\beta)$.
   // Decryption using $\mathsf{sk}_{1-\beta}$ omitted.
4. Compute $g^* = g^{a_{\beta,0}}.h^{a_{\beta,1}}.(g')^{a_{\beta,2}}.(h')^{a_{\beta,3}}$.
5. // If clause omitted — avoids dependency on $\mathsf{sk}_{1-\beta}$ and extraction trapdoor t.
6. Output g^*.

Figure 36: Let $\mathsf{crs} \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$ be in hiding mode. Additionally, let $(\mathsf{pk}_0, \mathsf{sk}_0), (\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathsf{FHE.Gen}(1^\lambda)$. Also, let $\{g_{\ell,\ell'}\}_{(\ell+\ell') \in [0,N]}$ be a set of group elements over the group $\mathcal{G}$ such that $\{g_{\ell,\ell'}\}_{(\ell+\ell') \in [0,N]} \leftarrow \mathcal{D}_{\mathsf{strong}-(N+1)-\mathsf{EDDH},\mathbb{G}}$ if $\beta' = 0$ and $\{g_{\ell,\ell'}\}_{(\ell+\ell') \in [0,N]} \leftarrow \mathcal{D}'_{\mathsf{strong}-(N+1)-\mathsf{EDDH},\mathbb{G}}$ otherwise. For each $\beta \in \{0, 1\}$, we define a deterministic circuit $\widehat{C}_{\mathsf{Ext,MMap},\beta}$ (which has the public keys pk_0 and pk_1, the secret key sk_β and the set $\{g_{\ell,\ell'}\}_{(\ell+\ell') \in [0,N]}$ hard-wired into it) as in Figure 36 below.

Lemma 6.5. *Hybrid-3 is computationally indistinguishable from Hybrid-2 assuming: (a) that NIZK satisfies* crs*-indistinguishability, perfect simulation-extractability under a binding* crs *and perfect zero-knowledge under a hiding* crs*, and (b) that piO satisfies probabilistic-iO security.*

The proof of this lemma is very similar to the proof of Lemma 4.4 and is hence not detailed.

Hybrid-4. In this hybrid, we change the manner in which the $(N+1)$-EDDH challenge encodings are generated. Suppose that the set of group elements $\{g_{\ell,\ell'}\}_{(\ell+\ell') \in [0,N]}$ embedded in the extraction circuit $\widehat{C}_{\mathsf{Ext,MMap},0,0}$ is distributed

as:

$$\{g_{\ell,\ell'}\}_{(\ell+\ell')\in[0,N]} = \left\{ g^{\gamma^{\ell+(N+1)\ell'}} \right\}_{(\ell+\ell')\in[0,N]}.$$

We switch the plaintext underlying the *second* FHE ciphertext in each challenge encoding as follows:

$$\begin{aligned} (\alpha, 0, 0, 0, \ldots, 0, i) &\rightarrow (0, (\alpha \cdot \gamma^{-1}), 0, 0, \ldots, 0, i) \\ (\alpha^{N+1}, 0, 0, 0, \ldots, 0, i) &\rightarrow (0, 0, (\alpha \cdot \gamma^{-1})^{N+1}, 0, \ldots, 0, i) \end{aligned}$$

Lemma 6.6. *Hybrid-4 is computationally indistinguishable from Hybrid-3 assuming that FHE is IND-CPA secure with respect to the public key-secret key pair* $(\mathsf{pk}_1, \mathsf{sk}_1)$.

The proof of this lemma is very similar to the proof of Lemma 4.5 and is hence not detailed.

Hybrid-5. In this hybrid, we change the public parameter pp as follows: compute $\mathsf{C}_{\mathsf{Ext,piO}} = \mathsf{piO.Obf}(\widehat{\mathsf{C}}_{\mathsf{Ext,MMap},0,1})$, where $\widehat{\mathsf{C}}_{\mathsf{Ext,MMap},0,1}$ is as defined in Figure 12.

Lemma 6.7. *Hybrid-5 is computationally indistinguishable from Hybrid-4 assuming that the strong* $(N+1)$*-EDDH assumption holds over the group* $\mathbb{G}$.

Proof. Note that the only difference between hybrids 4 and 5 is the manner in which the obfuscated extraction circuit $\mathsf{C}_{\mathsf{Ext,piO}}$ is generated as part of the public parameter.

In Hybrid-4, $\mathsf{C}_{\mathsf{Ext,piO}}$ is an obfuscation of the circuit $\widehat{\mathsf{C}}_{\mathsf{Ext,MMap},0,0}$, while in Hybrid-5, $\mathsf{C}_{\mathsf{Ext,piO}}$ is an obfuscation of the circuit $\widehat{\mathsf{C}}_{\mathsf{Ext,MMap},0,1}$ (where both circuits are as defined in Figure 12).

Now, observe that the only difference between the extraction circuits $\widehat{\mathsf{C}}_{\mathsf{Ext,MMap},0,0}$ and $\widehat{\mathsf{C}}_{\mathsf{Ext,MMap},0,1}$ is that the former embeds a set of group elements sampled from the distribution $\mathcal{D}_{\text{strong}-(N+1)-\text{EDDH},\mathbb{G}}$ over the group $\mathbb{G}$, while the latter embeds a set of group elements sampled from the distribution $\mathcal{D}'_{\text{strong}-(N+1)-\text{EDDH},\mathbb{G}}$ over the group $\mathbb{G}$.

It immediately follows that Hybrid-5 is computationally indistinguishable from Hybrid-4 assuming that the strong $(N+1)$-EDDH assumption holds over the group $\mathbb{G}$.

Hybrid-6. In this hybrid, we change the manner in which the $(N+1)$-EDDH challenge encodings are generated. Recall that in hybrid-5, the set of group elements $\{g_{\ell,\ell'}\}_{(\ell+\ell')\in[0,N]}$ in the extraction circuit $\widehat{C}_{\mathsf{Ext,MMap},0,1}$ is distributed as:

$$\{g_{\ell,\ell'}\}_{(\ell+\ell')\in[0,N]} = \left\{ g^{\gamma^{\ell}\cdot\delta^{\ell'}} \right\}_{(\ell+\ell')\in[0,N]}.$$

We switch the plaintext underlying the *second* FHE ciphertext in each challenge encoding as follows:

$$\begin{aligned} (0, (\alpha \cdot \gamma^{-1}), 0, 0, \ldots, 0, i) &\rightarrow (\alpha, 0, 0, 0, \ldots, 0, i) \\ (0, 0, (\alpha \cdot \gamma^{-1})^{N+1}, 0, \ldots, 0, i) &\rightarrow ((\alpha \cdot \gamma^{-1})^{N+1} \cdot \delta, 0, 0, 0, \ldots, 0, i) \end{aligned}$$

Lemma 6.8. *Hybrid-6 is computationally indistinguishable from Hybrid-5 assuming that FHE is IND-CPA secure with respect to the public key-secret key pair* $(\mathsf{pk}_1, \mathsf{sk}_1)$.

The proof of this lemma is very similar to the proof of Lemma 4.7 and is hence not detailed.

Hybrid-7. In this hybrid, we change the public parameter pp as follows: compute $\mathsf{C}_{\mathsf{Ext,piO}} = \mathsf{piO.Obf}(\widehat{\mathsf{C}}_{\mathsf{Ext,MMap},1,1})$, where $\widehat{\mathsf{C}}_{\mathsf{Ext,MMap},1,1}$ is as defined in Figure 36.

Lemma 6.9. *Hybrid-7 is computationally indistinguishable from Hybrid-6 assuming: (a) that NIZK satisfies* crs*-indistinguishability, perfect simulation-extractability under a binding* crs *and perfect zero-knowledge under a hiding* crs*, and (b) that piO satisfies probabilistic-iO security.*

The proof of this lemma is very similar to the proof of Lemma 4.8 and is hence not detailed.

Hybrid-8. In this hybrid, we change the manner in which the $(N+1)$-EDDH challenge encodings are generated. Recall that in hybrid-7, the set of group elements $\{g_{\ell,\ell'}\}_{(\ell+\ell')\in[0,N]}$ in the extraction circuit $\widehat{C}_{\mathsf{Ext,MMap},1,1}$ is distributed as:

$$\{g_{\ell,\ell'}\}_{(\ell+\ell')\in[0,N]} = \left\{ g^{\gamma^{\ell}\cdot\delta^{\ell'}} \right\}_{(\ell+\ell')\in[0,N]}.$$

We switch the plaintext underlying the *first* FHE ciphertext in each challenge encoding as follows:

$$\begin{aligned} (0,(\alpha\cdot\gamma^{-1}),0,0,\ldots,0,i) &\rightarrow (\alpha,0,0,0,\ldots,0,i) \\ (0,0,(\alpha\cdot\gamma^{-1})^{N+1},0,\ldots,0,i) &\rightarrow ((\alpha\cdot\gamma^{-1})^{N+1}\cdot\delta,0,0,0,\ldots,0,i) \end{aligned}$$

Lemma 6.10. *Hybrid-8 is computationally indistinguishable from Hybrid-7 assuming that FHE is IND-CPA secure with respect to the public key-secret key pair* $(\mathsf{pk}_0,\mathsf{sk}_0)$.

The proof of this lemma is very similar to the proof of Lemma 4.9 and is hence not detailed.

Hybrid-9. In this hybrid, we change the public parameter pp as follows:

1. Switch crs for the NIZK proof system to binding mode, as in the real scheme.
2. Switch $y \in \mathcal{X}$ to a uniform non-member instance, as in the real scheme.
3. Compute $\mathsf{C}_{\mathsf{Add,piO}} = \mathsf{piO.Obf}(\mathsf{C}_{\mathsf{Add,MMap}})$, as in the real scheme.
4. Compute $\mathsf{C}_{\mathsf{Inv,piO}} = \mathsf{piO.Obf}(\mathsf{C}_{\mathsf{Inv,MMap}})$, as in the real scheme.
5. Compute $\mathsf{C}_{\mathsf{Mult,piO}} = \mathsf{piO.Obf}(\mathsf{C}_{\mathsf{Mult,MMap}})$, as in the real scheme.
6. Compute $\mathsf{C}_{\mathsf{Ext,piO}} = \mathsf{piO.Obf}(\mathsf{C}_{\mathsf{Ext,MMap}})$, as in the real scheme.

Lemma 6.11. *Hybrid-9 is computationally indistinguishable from Hybrid-8 assuming: (a) that deciding subset membership with respect to* $\mathcal{L} \subset \mathcal{X}$ *is computationally hard, (b) that NIZK satisfies* crs*-indistinguishability, perfect simulation-extractability under a binding* crs *and perfect zero-knowledge under a hiding* crs*, and (c) that piO satisfies probabilistic-iO security.*

The proof of this lemma is very similar to the proof of Lemma 4.10 and is hence not detailed.

Finally, Hybrid-9 is identical to the game where the adversary is given a random invalid $(N+1)$-EDDH instance generated as per the real scheme. This completes the proof of Theorem 6.1.

6.8 Power-DDH Implies Strong EDDH

In this subsection, we prove the following lemma:

Lemma 6.12. *The "strong"-$(N+1)$-EDDH assumption over a group $\mathbb{G}$ for $N \geq 1$ is implied by the power-DDH assumption over the group $\mathbb{G}$.*

Proof. Recall that the strong-$(N+1)$-EDDH assumption states that letting $g \leftarrow \mathbb{G}$ and $\gamma, \delta \leftarrow \mathbb{Z}_q^*$, the following indistinguishability holds:

$$\left(\left\{g^{\gamma^{\ell+(N+1)\ell'}}\right\}_{(\ell+\ell')\in[0,N]}\right) \stackrel{c}{\approx} \left(\left\{g^{\gamma^{\ell}\cdot\delta^{\ell'}}\right\}_{(\ell+\ell')\in[0,N]}\right).$$

Also recall that the power DDH assumption states that the following indistinguishability holds for any $t = \mathrm{poly}(\lambda)$:

$$\left(g, g^a, g^{a^2}, \ldots, g^{a^t}\right) \quad \text{or} \quad \left(g, g^{r_1}, g^{r_2}, \ldots, g^{r_t}\right),$$

where $g \leftarrow \mathbb{G}$ and $a, r_1, r_2, \ldots, r_t \leftarrow \mathbb{Z}_q$. Hence, by the power-DDH assumption, we have:

$$\left(\left\{g^{\gamma^{\ell+(N+1)\ell'}}\right\}_{(\ell+\ell')\in[0,N]}\right) \stackrel{c}{\approx} \left(\left\{g^{r_{\ell,\ell'}}\right\}_{(\ell+\ell')\in[0,N]}\right),$$

where each $r_{\ell,\ell'} \leftarrow \mathbb{Z}_q$. Also, by the power-DDH assumption, we have:

$$\begin{aligned}\left(\left\{g^{\gamma^{\ell}\cdot\delta^{\ell'}}\right\}_{(\ell+\ell')\in[0,N]}\right) &\stackrel{c}{\approx} \left(\left\{g^{r_{\ell}\cdot\delta^{\ell'}}\right\}_{(\ell+\ell')\in[0,N]}\right)\\ &\stackrel{c}{\approx} \left(\left\{g^{r_{\ell}\cdot r'_{\ell'}}\right\}_{(\ell+\ell')\in[0,N]}\right).\end{aligned}$$

where each $r_\ell, r'_{\ell'} \leftarrow \mathbb{Z}_q$. Finally, by the DDH assumption (which is implied by the power DDH assumption), we have:

$$\left(\left\{g^{r_{\ell}\cdot r'_{\ell'}}\right\}_{(\ell+\ell')\in[0,N]}\right) \stackrel{c}{\approx} \left(\left\{g^{r_{\ell,\ell'}}\right\}_{(\ell+\ell')\in[0,N]}\right),$$

where each $r_{\ell,\ell'} \leftarrow \mathbb{Z}_q$. This completes the proof of Lemma 6.12. □

References

[AB15] Benny Applebaum and Zvika Brakerski. Obfuscating circuits via composite-order graded encoding. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 528–556. Springer, Heidelberg, March 2015.

[ABG+13] Prabhanjan Ananth, Dan Boneh, Sanjam Garg, Amit Sahai, and Mark Zhandry. Differing-inputs obfuscation and applications. Cryptology ePrint Archive, Report 2013/689, 2013. https://eprint.iacr.org/2013/689.

[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.

[Agr19] Shweta Agrawal. Indistinguishability obfuscation without multilinear maps: New methods for bootstrapping and instantiation. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part I*, volume 11476 of *LNCS*, pages 191–225. Springer, Heidelberg, May 2019.

[AJ15] Prabhanjan Ananth and Abhishek Jain. Indistinguishability obfuscation from compact functional encryption. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 308–326. Springer, Heidelberg, August 2015.

[AJL+19] Prabhanjan Ananth, Aayush Jain, Huijia Lin, Christian Matt, and Amit Sahai. Indistinguishability obfuscation without multilinear maps: New paradigms via low degree weak pseudorandomness and security amplification. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 284–332. Springer, Heidelberg, August 2019.

[AM18] Shweta Agrawal and Monosij Maitra. FE and iO for turing machines from minimal assumptions. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018, Part II*, volume 11240 of *LNCS*, pages 473–512. Springer, Heidelberg, November 2018.

[AMP20] Navid Alamati, Hart Montgomery, and Sikhar Patranabis. Ring key-homomorphic weak prfs and applications. Cryptology ePrint Archive, Report 2020/606, 2020.

[AP20] Shweta Agrawal and Alice Pellet-Mary. Indistinguishability obfuscation without maps: Attacks and fixes for noisy linear FE. In Vincent Rijmen and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, LNCS, pages 110–140. Springer, Heidelberg, May 2020.

[AS15] Gilad Asharov and Gil Segev. Limits on the power of indistinguishability obfuscation and functional encryption. In Venkatesan Guruswami, editor, *56th FOCS*, pages 191–209. IEEE Computer Society Press, October 2015.

[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 *LNCS*, pages 152–181. Springer, Heidelberg, April / May 2017.

[BBKK18] Boaz Barak, Zvika Brakerski, Ilan Komargodski, and Pravesh K. Kothari. Limits on low-degree pseudorandom generators (or: Sum-of-squares meets program obfuscation). In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 649–679. Springer, Heidelberg, April / May 2018.

[BCP14] Elette Boyle, Kai-Min Chung, and Rafael Pass. On extractability obfuscation. In Yehuda Lindell, editor, *TCC 2014*, volume 8349 of *LNCS*, pages 52–73. Springer, Heidelberg, February 2014.

[BDGM20] Zvika Brakerski, Nico Döttling, Sanjam Garg, and Giulio Malavolta. Candidate iO from homomorphic encryption schemes. In Vincent Rijmen and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, LNCS, pages 79–109. Springer, Heidelberg, May 2020.

[BGI+01] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 1–18. Springer, Heidelberg, August 2001.

[BMSZ15] Saikrishna Badrinarayanan, Eric Miles, Amit Sahai, and Mark Zhandry. Post-zeroizing obfuscation: The case of evasive circuits. Cryptology ePrint Archive, Report 2015/167, 2015. http://eprint.iacr.org/2015/167.

[BP15] Nir Bitansky and Omer Paneth. ZAPs and non-interactive witness indistinguishability from indistinguishability obfuscation. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 401–427. Springer, Heidelberg, March 2015.

[BSW16] Mihir Bellare, Igors Stepanovs, and Brent Waters. New negative results on differing-inputs obfuscation. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 792–821. Springer, Heidelberg, May 2016.

[BV15] Nir Bitansky and Vinod Vaikuntanathan. Indistinguishability obfuscation from functional encryption. In Venkatesan Guruswami, editor, *56th FOCS*, pages 171–190. IEEE Computer Society Press, October 2015.

[BZ14] Dan Boneh and Mark Zhandry. Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 480–499. Springer, Heidelberg, August 2014.

[CGH17] Yilei Chen, Craig Gentry, and Shai Halevi. Cryptanalyses of candidate branching program obfuscators. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part III*, volume 10212 of *LNCS*, pages 278–307. Springer, Heidelberg, April / May 2017.

[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.

[CKWZ13] Seung Geol Choi, Jonathan Katz, Hoeteck Wee, and Hong-Sheng Zhou. Efficient, adaptively secure, and composable oblivious transfer with a single, global CRS. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *PKC 2013*, volume 7778 of *LNCS*, pages 73–88. Springer, Heidelberg, February / March 2013.

[CLLT16] Jean-Sébastien Coron, Moon Sung Lee, Tancrède Lepoint, and Mehdi Tibouchi. Cryptanalysis of GGH15 multilinear maps. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part II*, volume 9815 of *LNCS*, pages 607–628. Springer, Heidelberg, August 2016.

[CLT13] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. Practical multilinear maps over the integers. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 476–493. Springer, Heidelberg, August 2013.

[CLT15] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. New multilinear maps over the integers. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 267–286. Springer, Heidelberg, August 2015.

[CLTV15] Ran Canetti, Huijia Lin, Stefano Tessaro, and Vinod Vaikuntanathan. Obfuscation of probabilistic circuits and applications. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 468–497. Springer, Heidelberg, March 2015.

[Dam88] Ivan Damgård. Collision free hash functions and public key signature schemes. In David Chaum and Wyn L. Price, editors, *EUROCRYPT'87*, volume 304 of *LNCS*, pages 203–216. Springer, Heidelberg, April 1988.

[EHK+13] Alex Escala, Gottfried Herold, Eike Kiltz, Carla Ràfols, and Jorge Villar. An algebraic framework for Diffie-Hellman assumptions. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 129–147. Springer, Heidelberg, August 2013.

[FHHL18] Pooya Farshim, Julia Hesse, Dennis Hofheinz, and Enrique Larraia. Graded encoding schemes from obfuscation. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018, Part II*, volume 10770 of *LNCS*, pages 371–400. Springer, Heidelberg, March 2018.

[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.

[GGH15] Craig Gentry, Sergey Gorbunov, and Shai Halevi. Graph-induced multilinear maps from lattices. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 498–527. Springer, Heidelberg, March 2015.

[GGHR14] Sanjam Garg, Craig Gentry, Shai Halevi, and Mariana Raykova. Two-round secure MPC from indistinguishability obfuscation. In Yehuda Lindell, editor, *TCC 2014*, volume 8349 of *LNCS*, pages 74–94. Springer, Heidelberg, February 2014.

[GGHW17] Sanjam Garg, Craig Gentry, Shai Halevi, and Daniel Wichs. On the implausibility of differing-inputs obfuscation and extractable witness encryption with auxiliary input. *Algorithmica*, 79(4):1353–1373, 2017.

[GLSW15] Craig Gentry, Allison Bishop Lewko, Amit Sahai, and Brent Waters. Indistinguishability obfuscation from the multilinear subgroup elimination assumption. In Venkatesan Guruswami, editor, *56th FOCS*, pages 151–170. IEEE Computer Society Press, October 2015.

[GMM+16] Sanjam Garg, Eric Miles, Pratyay Mukherjee, Amit Sahai, Akshayaram Srinivasan, and Mark Zhandry. Secure obfuscation in a weak multilinear map model. Cryptology ePrint Archive, Report 2016/817, 2016. http://eprint.iacr.org/2016/817.

[GPSZ17] Sanjam Garg, Omkant Pandey, Akshayaram Srinivasan, and Mark Zhandry. Breaking the sub-exponential barrier in obfustopia. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part III*, volume 10212 of *LNCS*, pages 156–181. Springer, Heidelberg, April / May 2017.

[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.

[HB15] Máté Horváth and Levente Buttyán. The birth of cryptographic obfuscation – a survey. Cryptology ePrint Archive, Report 2015/412, 2015. https://eprint.iacr.org/2015/412.

[HJ16] Yupu Hu and Huiwen Jia. Cryptanalysis of GGH map. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part I*, volume 9665 of *LNCS*, pages 537–565. Springer, Heidelberg, May 2016.

[HSW14] Susan Hohenberger, Amit Sahai, and Brent Waters. Replacing a random oracle: Full domain hash from indistinguishability obfuscation. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 201–220. Springer, Heidelberg, May 2014.

[JLMS19] Aayush Jain, Huijia Lin, Christian Matt, and Amit Sahai. How to leverage hardness of constant-degree expanding polynomials overa $\mathbb{R}$ to build $i\mathcal{O}$. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part I*, volume 11476 of *LNCS*, pages 251–281. Springer, Heidelberg, May 2019.

[Lin16] Huijia Lin. Indistinguishability obfuscation from constant-degree graded encoding schemes. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part I*, volume 9665 of *LNCS*, pages 28–57. Springer, Heidelberg, May 2016.

[LT17] Huijia Lin and Stefano Tessaro. Indistinguishability obfuscation from trilinear maps and block-wise local PRGs. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 630–660. Springer, Heidelberg, August 2017.

[LV16] Huijia Lin and Vinod Vaikuntanathan. Indistinguishability obfuscation from DDH-like assumptions on constant-degree graded encodings. In Irit Dinur, editor, *57th FOCS*, pages 11–20. IEEE Computer Society Press, October 2016.

[LV17] Alex Lombardi and Vinod Vaikuntanathan. Limits on the locality of pseudorandom generators and applications to indistinguishability obfuscation. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part I*, volume 10677 of *LNCS*, pages 119–137. Springer, Heidelberg, November 2017.

[MSZ16] Eric Miles, Amit Sahai, and Mark Zhandry. Annihilation attacks for multilinear maps: Cryptanalysis of indistinguishability obfuscation over GGH13. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part II*, volume 9815 of *LNCS*, pages 629–658. Springer, Heidelberg, August 2016.

[MZ18] Fermi Ma and Mark Zhandry. The MMap strikes back: Obfuscation and new multilinear maps immune to CLT13 zeroizing attacks. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018, Part II*, volume 11240 of *LNCS*, pages 513–543. Springer, Heidelberg, November 2018.

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

[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.

[Wat15] Brent Waters. A punctured programming approach to adaptively secure functional encryption. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 678–697. Springer, Heidelberg, August 2015.

[YYHK14] Takashi Yamakawa, Shota Yamada, Goichiro Hanaoka, and Noboru Kunihiro. Self-bilinear map on unknown order groups from indistinguishability obfuscation and its applications. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part II*, volume 8617 of *LNCS*, pages 90–107. Springer, Heidelberg, August 2014.

[Zim15] Joe Zimmerman. How to obfuscate programs directly. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 439–467. Springer, Heidelberg, April 2015.