Stronger Multilinear Maps from Indistinguishability Obfuscation [*]

Navid Alamati[†] Hart Montgomery [‡] Sikhar Patranabis[§]

December 29, 2020

Abstract

We show how to construct new multilinear maps from subexponentially secure indistinguishability obfuscation (iO) and (essentially) 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 and 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.

Coupled with the recent work of Jain *et al.*, which shows how to build iO from well-founded assumptions, and the recent work of Brakerski *et al.*, which shows how to build iO from circular security assumptions, our work can be used to also build strong multilinear maps from such assumptions. This solves another long-standing open problem. Moreover, 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.

[*]We thank the anonymous reviewers of TCC 2020 for pointing out a technical problem in an earlier version of the paper. The current version fixes this problem.

[†]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].

The story of realizing iO from concrete assumptions began with multilinear maps: 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 were proposed subsequently [CLT13, GGH15, Zim15, AB15].

However, many of these candidate constructions of multilinear maps were cryptanalyzed, 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 seemed to convince the cryptographic community to focus on alternative ways of constructing iO. One such approach that has 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]

But these attacks on FE-based iO constructions seemed to be temporary setbacks rather than a fundamental barrier. Almost all 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 more recent result of Brakerski *et al.* [BDGM20a] shows how to build iO from a new primitive called *split FHE*. Their construction is based on plausible assumptions, but relies on a heuristic security argument. While some of these new assumptions are not yet well-understood, they appear to be more and more "standard-looking." It is true that some of these recent results have been cryptanalyzed [AP20], but, unlike multilinear map constructions, most have not been broken.

Recent iO Results. Very recently, there have been a collection of exciting breakthrough results on new iO constructions. In particular, Jain *et al.* [JLS20] proposed a new construction of iO from well-founded assumptions, which includes the learning with errors (LWE) assumption [Reg05], the symmetric external Diffie-Hellman assumption (SXDH) [BGdMM05], a version of the learning parity with noise (LPN) assumption [Ale03] over a large modulus, and the existence of a boolean PRG in NC^0 with superlinear stretch. Like much of the previous work, this construction goes through functional encryption and does not use multilinear maps at all. This is the first work to build iO

[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.

entirely from assumptions that have been previously studied by the cryptography and theory communities and has led the cryptographic community to generally believe that iO can be realized.

Subsequently, Gay and Pass [GP20] proposed a variant of the [BDGM20a] scheme that only relies on the LWE assumption, the decisional composite residuosity assumption [DJ01], and a circular security assumption on the Gentry-Sahai-Waters FHE scheme [GSW13] and the Damgard-Jurik encryption scheme [DJ01]. In a follow-up work, Brakerski *et al.* showed how to construct iO while relying *only* on the circular-security of certain LWE-based schemes [BDGM20b].

These recent breakthrough results seem to point towards the fact that we can, in fact, realize iO–and maybe even from fully standard assumptions (i.e. just plain LWE) in the future. Unfortunately, they say nothing about the existence of multilinear maps.

1.1 Multilinear Maps and iO

The reported attacks on multilinear maps/graded encodings, together with the exciting new constructions of iO that do not involve multilinear maps, 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 with which they are endowed: multilinear maps with hardness assumptions over the source groups, and multilinear maps with hardness assumptions over the target groups.

"Source" and "Target" Group Assumptions. Suppose we consider an asymmetric multilinear map $e : \mathbb{G}_1 \times \ldots \times \mathbb{G}_\ell \to \mathbb{G}_T$ 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.[4]

Next, let's consider a symmetric multilinear map $e : \mathbb{G}^\ell \to \mathbb{G}_T$ 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 $e\left(g, \ldots, g\right)^{a_1 \cdots a_\ell a_{\ell+1}}$ 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. Lin and Vaikuntanathan [LV16] and Lin [Lin17] also showed constructions of iO from multilinear maps with source group assumptions (and some other relatively standard assumptions). 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. Even the functional encryption-based constructions of iO have a similar drawback and must assume subexponential hardness at some point in the reduction. We refer the reader to [GLSW15] for a detailed discussion on this issue.

[4]Note that SXDH can only be valid in an asymmetric multilinear map. Furthermore, there are subtleties when definining SXDH over high-degree multilinear maps, and not all definitions may be equivalent to what we outline here. We explain this in detail in the body of the paper.

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 richer applications such as multi-party 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 multilinear DDH assumption holds. 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 obviously implies that multilinear maps are at least as strong as iO, but the lack of constructions of "strong" multilinear maps with source-group hardness also apparently suggests that multilinear maps with such source group assumptions could be 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 (even piO) is seemingly insufficient here since the obfuscated programs do not have identical outputs on known inputs.

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 that are at least strong enough to imply a CRHF.

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 for X-Ind samplers (implied by subexponentially secure standard iO and subexponentially secure puncturable PRFs in $\mathcal{NC}^1$).
- Fully homomorphic encryption (FHE) with message space $\mathbb{Z}_q$ for some (large) prime q with perfect decryption correctness and well-distributed homomorphic evaluation outputs.[5]
- 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.
- An *asymmetric* multilinear map that is joint-SXDH-hard,[6] assuming additionally the existence of any DDH-hard group of prime order.
- An *asymmetric* multilinear map that is exponent-DDH-hard, assuming additionally the existence of any exponent-DDH-hard group of prime order.
- An n-degree *symmetric* multilinear map that is $(n+1)$-exponent-DDH-hard, assuming additionally the existence of any power-DDH-hard group of prime order.

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 $\mathcal{NC}^1$ 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 achieve 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.

[5]By well-distributed, we mean that the output of homomorphic evaluation of a function is identically distributed as the output of the encryption algorithm on the same function of the message. This kind of FHE is not known from the LWE assumption, but can be constructed from iO and standard assumptions [CLTV15].

[6]Our definition of j-SXDH is slightly different (although, seemingly equivalent in practice) to the standard definition [LV16].

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).

Using the very recently proposed constructions of iO from well-founded assumptions [JLS20] and circular security assumptions [GP20, BDGM20b] in conjunction with our work, one can also build multilinear maps from well-founded assumptions, which seemingly answers an almost twenty-year old question on the existence of multilinear maps beyond degree 2 [BS03].

Simpler Constructions from MMaps. One benefit of our work is simpler feasibility results for primitives that are only known from iO or multilinear maps. As an example, consider multi-party noninteractive key exchange (NIKE). Boneh and Silverberg's construction of NIKE from multilinear maps [BS03] is very simple and has an almost immediate proof of security. On the other hand, Boneh and Zhandry's construction of NIKE from iO [BZ14], while elegant, is quite complicated and the proof is not so simple. In many cases, cryptosystems are simpler and easier to build from multilinear map assumptions than iO.

Our construction of multilinear maps from iO is not particularly efficient. But known constructions of iO are not efficient at all either, and iO is mostly used at this point for feasibility results on complex cryptoprimitives. For feasibility results–where we are only trying to prove the existence of some primitive from some assumption(s), and mostly ignoring efficiency–our construction of multilinear maps still might be incredibly useful. For new constructions, we could prove the security of the construction based on a multilinear map and base security on good assumptions through our reduction from iO in this paper. Later, we could try to build a more efficient scheme. As we have stated before, almost all of the results on applications of iO have focused on feasibility rather than efficiency, so, while we do think efficiency of iO is an interesting problem going forward, we do not consider the lack of efficiency in our construction a glaring weakness.

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. We can even "bootstrap" symmetric multilinear maps into asymmetric ones, and vice versa. Our work implies that many of the (seemingly very different) multilinear maps that are powerful enough to imply iO are, up to subexponential security reductions and standard assumptions, equivalent in a computational hardness sense. 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 an *unbounded* degree of multilinearity (or even if such maps exist). 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.

Finally, attempting to build multilinear maps directly from some of the assumptions used to build iO in a recent line of works [JLS20, GP20, BDGM20b] seems to be an interesting and potentially promising open problem.

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 points of our construction are 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. The reader may refer to 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 an SXDH-hard asymmetric multilinear map (MMap)/graded encoding from iO and other cryptographic assumptions. Our proofs are quite involved, so we cannot mention all of the steps or techniques here. In Section 6 we present our full proof of SXDH-hardness with a detailed outline of all of the steps. Rather than mimic the same presentation here, we describe the proof ideas in a more intuitive manner. While this description does not linearly follow how the proof is actually presented in the body of the paper, it captures the main technical ideas.

We will start by sketching out what the encodings of our multilinear map look like, and then explain our circuits for multilinear map operations.

Non-Unique Representations. We begin by noting that our encodings are *not unique*: there will be (potentially) multiple ways to represent and encode some plaintext element $\alpha \in \mathbb{Z}_q$ (q being a prime with $O(\lambda)$ bits, where λ is the security parameter). More precisely, we will use a "slotted" representation of α for our encodings, the exact form of which will depend on the assumption that we are trying to prove. For our construction of an SXDH-hard asymmetric MMap, we will represent an element α as a four-tuple of the form $(\alpha_0, \alpha_1, \alpha_2, \alpha_3)$ with the restriction that, for some fixed $a, b, c \in \mathbb{Z}_q$ (sampled at setup), we have

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

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.

Encoding Structure. Each encoding in our construction consists of two FHE ciphertexts that are encryptions of the underlyig plaintext element $\alpha \in \mathbb{Z}_q$ and the "level" of the encoding 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" of the encoding which we denote $\mathbf{i}$, and two NIZK proofs π_0 and π_1 that prove, in a sense, the encoding is valid (we will explain these in more detail later). We can express this as:

$$\mathsf{Encode}(\alpha, \mathbf{i}) = \left(\mathsf{FHE.Enc}_{\mathsf{pk}_0}(\alpha, \mathbf{i}), \mathsf{FHE.Enc}_{\mathsf{pk}_1}(\alpha, \mathbf{i}), \mathbf{i}, \pi_0, \pi_1\right).$$

We have already explained the representation(s) that may be used when encoding an element α. The description of the level of an encoding in our SXDH-hard multilinear map is a binary vector $\mathbf{i} \in \{0,1\}^n$, where n is the degree of the multilinearity of the map. Top-level encodings corresponding to each of the groups $\mathbb{G}_1, ..., \mathbb{G}_n$ are denoted by $\mathbf{i}$s with one non-zero entry. Addition preserves the level set, so $\mathbf{i}$ remains unchanged, while multiplication (which can only be done between level sets that have an empty intersection) of two elements with level sets $\mathbf{i}_1$ and $\mathbf{i}_2$ results in a new level set $\mathbf{i}' = \mathbf{i}_1 + \mathbf{i}_2$. Other multilinear maps (e.g. symmetric constructions) may have simpler level descriptions.

Normal and Oblique Representations. Before we explain our proofs π_0 and π_1, we need to explain more about the nature of our encodings. When encoding an element α, depending on whether we use only the α_0 component or all four components $(\alpha_0, \alpha_1, \alpha_2, \alpha_3)$, we classify the encoding representation into "normal", "partially oblique" and "oblique". The tuple $\alpha = (\alpha_0, \alpha_1, \alpha_2, \alpha_3)$ is said to be in "normal form" if

$$(\alpha_0, \alpha_1, \alpha_2, \alpha_3) = (\alpha^*, 0, 0, 0)$$

for some $\alpha^* \in \mathbb{Z}_q$. Otherwise, it is said to be in "oblique form". Depending on the forms of the tuples underlying the FHE ciphertexts in the encoding, we classify an encoding into one of three representations:

- **Normal representation:** Both FHE ciphertexts encrypt tuples that are in normal form.
- **Partially oblique representation:** Exactly one of the FHE ciphertexts encrypts a tuple that is in normal form, while the other encrypts a tuple that is in oblique form.
- **Oblique representation:** Both FHE ciphertexts encrypt tuples that are in oblique form.

Consistency of Encodings. We also associate with any given encoding a property called "consistency", which basically captures that both FHE ciphertexts correspond to encryptions of the same plaintext element. In particular, if the FHE ciphertexts in an encoding encrypt tuples of the form $(\alpha_0, \alpha_1, \alpha_2, \alpha_3)$ and $(\beta_0, \beta_1, \beta_2, \beta_3)$, then the endcoding is "consistent" with respect to the fixed elements $a, b, c \in \mathbb{Z}_q$ if the following condition is satisfied:

$$\alpha_0 + a \cdot \alpha_1 + b \cdot \alpha_2 + c \cdot \alpha_3 = \beta_0 + a \cdot \beta_1 + b \cdot \beta_2 + c \cdot \beta_3 \mod q.$$

Note that an encoding can be consistent irrespective of its representation (normal, oblique or partially oblique).

Zero Knowledge Proofs We are finally in position to discuss our zero knowledge proofs π_0 and π_1. It is essential to the construction that we use a dual-mode, simulation-extractable non-interactive zero knowledge argument system (e.g. Groth-Sahai [GS08]). In particular, we need a NIZK proof system that is perfectly sound and extractable in the binding mode, and perfectly witness indistinguishable and perfectly zero-knowledge in the hiding mode. In our 'real' construction, all of our proofs will be done in the binding mode. We will use hiding mode for certain hybrid arguments.

The Proof π_1. Since it is simpler, we will start by discussing π_1. Informally, π_1 is a proof of consistency. It proves that both FHE ciphertexts encode the same α, as per the definition of consistency presented earlier. However, the actual proof is a bit more complicated. In particular, the proof π_1 is a proof that *either* the FHE ciphertexts are encoded consistently *or* the prover has knowledge of a unique witness wit_y associated with an instance y of some language $\mathcal{L}$ with hard membership. Informally, this language has the property that if we sample y randomly from $\mathcal{L}$, it is computationally hard to infer if $y \in \mathcal{L}$ or not, but each y that is a "yes" instance has a unique membership witness wit_y (we define these languages formally in our preliminaries).

In our actual scheme, we choose to use a non-accepting y value, so this 'or' branch can never be satisfied due to the perfect soundness in the binding mode of our NIZK proof system. However, this extra 'or' branch is necessary for our proofs and is activated in some of our hybrid schemes and circuits. We explain the intuition behind these kinds of proofs in more detail later in this overview, as well as in the main body of the paper.

The Proof π_0. Our proof π_0 is a little bit more complicated. It uses $2n$ additional instances of the hard language $\mathcal{L}$, which we denote $z_{j,b}$ for $j \in [n]$ and $b \in \{0,1\}$. Informally, in π_0 we prove that both FHE ciphertexts encrypt the same level-set $\mathbf{i} = (\mathbf{i}_1, \ldots, \mathbf{i}_n)$ as described "in the clear" in the encoding, and the fact that (at least) one of the following conditions holds:

- *Either* both FHE ciphertexts encode the same α in *normal representation* (i.e., the corresponding plaintexts are identical to each other).
- *Or* there exists some $j \in [1, n]$ such that $z_{j,0} \in \mathcal{L}$ *and* $\mathbf{i}_j = 0$.
- *Or* there exists some $j \in [1, n]$ such that $z_{j,1} \in \mathcal{L}$ *and* $\mathbf{i}_j = 1$.

So, to summarize, π_0 proves that *either* an encoding is in the normal form *or* that the prover possesses a unique membership witness corresponding to an instance of the hard language $\mathcal{L}$. At a first glance, using $2n$ instances of $\mathcal{L}$ might seem like overkill (and, in fact, our first attempt at a construction only used one instance). However, using $2n$ instances of $\mathcal{L}$ helps us to have fine-grained control over what particular types of oblique encoding are allowed. For instance, if we set $z_{1,1}$ to be an "yes" instance of $\mathcal{L}$ and all other zs to be "no" instances (and put our proofs in binding mode), then our scheme will effectively only allow oblique encodings for encodings that "include" level 1 (i.e. the first group of the multilinear map, such that $\mathbf{i}_1 = 1$). As we will show later, this is crucial to our proof.

As before, these alternative 'or' branches never get used in our actual construction. They only are active in hybrid stages of our proof. All of our encodings in the actual construction will be in normal form.

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 indistinguishability obfuscations of circuits that broadly adhere to the following strategy:

- Verify the proofs π_0 and π_1 of both encodings to make sure they are valid (if not, abort).
- Use the FHE secret keys to decrypt the ciphertexts and retrieve the underlying input plaintexts.
- Perform the desired operation over the input plaintexts and create a valid encoding of the output plaintext at the appropriate level.

In some of our proof hybrids, we will want to "forget" some of the FHE secret keys and switch to computing addition, inversion, and multiplication homomorphically (without decryption). At a high level, such switches will allow us to transform one or more encodings from normal to oblique representation and vice versa. These steps will, of course, involve using the 'or' branches of our NIZK proofs. At the same time, these steps will also require the ability to compute the MMap operations directly over the FHE ciphertexts in the input encoding(s), without the knowledge of the corresponding secret keys. This is precisely why we need to use FHE (rather than some simpler form of encryption).

Multiplying Oblique Encodings. It is important to note here that multiplication of two encodings that are both in the oblique representation is impossible to compute unless the elements $a, b, c \in \mathbb{Z}_q$ used for the oblique representation are explicitly hardwired into the obfuscated multiplication circuit. This is the case for our 'real' scheme.

However, for reasons relevant to the proof of SXDH, in some of our hybrid arguments, we will want to not publish a, b, and c, and instead give out a tuple $\left(g, g^a, g^b, g^c\right)$. This means that we cannot multiply two elements in oblique form. However, in such hybrid arguments, we ensure that it is impossible to ever multiply two encodings that are both in oblique form. We expand more on this subsequently.

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

- Verify the proofs π_0 and π_1 to make sure the encoding is constructed correctly (if not, abort).
- Use the FHE secret keys to decrypt (at least one of) the ciphertexts and recover $\alpha = (\alpha_0, \alpha_1, \alpha_2, \alpha_3)$.
- Output $g^{\alpha_0 + a\alpha_1 + b\alpha_2 + c\alpha_3}$ 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]. Furthermore, we only (technically) need one of the two FHE secret keys in order to successfully extract. This follows from our proof of consistency π_1, which proves that the ciphertexts encode the same α. Looking ahead, we will exploit this in our reduction.

Comparison with Previous Works. We present a high-level comparison of our construction with those in [AFH$^+$16] and [FHHL18]. In these papers, the authors also encode elements using two FHE ciphertexts and use a zero knowledge proof (that they can relax by using a witness to an instance of a hard language problem in an 'or'-style proof, like we do) to argue that the proofs are consistent.

Our work here can be seen as a much more complicated generalization of their approach. The core differences between our construction and these previous works stem from 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. This generalization plays a crucial rule in achieving MMap constructions with stronger (source group) assumptions, as compared to the previous works. However, our more general encoding strategy also leads to additional requirements in the scheme: in particular, we must use two (more complicated) zero-knowledge proofs π_0, π_1 instead of a single zero-knowledge proof as in [AFH$^+$16] and [FHHL18]. Our proofs of security are also substantially longer and more complicated than those of these previous works.

In another prior work, Agrikola and Hofheinz [AH18] used a generalization of the encoding strategies in [AFH$^+$16] and [FHHL18], albeit towards a different goal - to construct a mathematical group in which an interactive variant of the very general Uber assumption holds. While their encoding strategies are conceptually similar to ours at a high level, the core techniques differ significantly.

2.2 Proof Intuition

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 normal representation to oblique representation 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 normal 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 will start by explaining our core proof goal at a high level. Then we will explain how some of our techniques for completing the proof work.

Overall Goal: Embedding a DDH Challenge. The core idea behind our proof is to embed a DDH challenge in our encodings: the "real" SXDH distribution will correspond to a "real" DDH term embedded in our encodings, and the "random" SXDH distribution will correspond to a "random" DDH term embedded. How exactly this works is a little bit complicated, and necessitates our use of oblique encodings to switch between "real" and "random" SXDH challenge encodings. Recall that the SXDH assumption requires that the following indistinguishability holds for a "top-level" level-set $\mathbf{i}$:[7]

$$(\mathsf{Encode}(\alpha, \mathbf{i}), \mathsf{Encode}(\beta, \mathbf{i}), \mathsf{Encode}(\alpha \cdot \beta, \mathbf{i})) \overset{c}{\approx} (\mathsf{Encode}(\alpha, \mathbf{i}), \mathsf{Encode}(\beta, \mathbf{i}), \mathsf{Encode}(\gamma, \mathbf{i})),$$

where $\alpha, \beta, \gamma \leftarrow \mathbb{Z}_q$. In our SXDH-hard MMap construction, we encode an element α as a tuple of the form $(\alpha_0, \alpha_1, \alpha_2, \alpha_3)$ with the restriction that for some fixed $a, b, c \in \mathbb{Z}_q$,

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

In our construction we consider two kinds of possible ways of representing an element α - the normal representation and the oblique representation. 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 an oblique representation, all terms may be arbitrary as long as the total sum is consistent. In hybrid arguments, we use the oblique encoding to help us prove security.

How might we do this? We can use the exponents a, b, and c to help here by directly tying them to the SXDH (and corresponding DDH) challenge. In particular, we will use these elements to encode a challenge that can either be "real" or "random" depending on the choices we make. More precisely, we can sample a and b uniformly from $\mathbb{Z}_q$, and, in the "real" case, set $c = ab$, and in the random case, sample $c \leftarrow \mathbb{Z}_q$. Our challenge elements will quite literally be $\mathsf{Encode}(a, \mathbf{i})$, $\mathsf{Encode}(b, \mathbf{i})$, and $\mathsf{Encode}(c, \mathbf{i})$ where $\mathbf{i}$ denotes the level-set at which we want to encode the challenge.

Note that this distribution of encodings is correct for an SXDH challenge. Moreover, we can use either one of the following representations for the challenge encodings:

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

Of course, an SXDH adversary should not be able to tell whether $c = ab$ or c is random. If we give out these terms in the clear (even to the obfuscator!), then proving security becomes difficult. However, it turns out we do not have to give a, b, and c out directly: we can give them out in the form of a DDH tuple (g, g^a, g^b, g^c) and work with the oblique encodings, which, as we showed above, are themselves independent of a, b, and c (even if these are needed for the evaluation). We can continue to compute (almost) all of our multlinear map algorithms using just this DDH tuple, and we can argue that, since the outputs are the same whether or not we use the DDH tuple or the terms a, b, and c in the clear for our multilinear map algorithms, the obfuscated programs for our circuits can "forget" a, b, and c themselves and use the DDH tuple at this step without any adversary being able to tell. This allows us to show that any adversary that can break this (crucial) step of our proof of SXDH can be used to break DDH. We elaborate on this below.

Group Embedding. Working with the tuple (g, g^a, g^b, g^c), rather than the exponents a, b, and c themselves, is a little bit tricky. However, we can (mostly) compute all of the required algorithms, which we show next.

Addition and Inversion. To begin, observe that addition and inversion of encodings do not require the knowledge of the exponents 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 group to compute encodings in the following way:

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

To see why this works, note that

$$g^{x_0 + ax_1 + bx_2 + cx_3} g^{y_0 + ay_1 + by_2 + cy_3} = g^{(x_0 + y_0) + a(x_1 + y_1) + b(x_2 + y_2) + c(x_3 + y_3)}.$$

Extraction and Zero-Testing. Extraction and zero-testing are also relatively simple to handle given only the DDH tuple of (g, g^a, g^b, g^c) instead of a, b, and c. Given an encoding of $x = (x_0, x_1, x_2, x_3)$, the extraction circuit outputs

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

[7]Some SXDH definitions require this level-set to be a *top-level* "group" (i.e. $\mathbf{i}_j = 1$ for some j and $\mathbf{i}_k = 0$ for all $k \neq j$), others require that it hold for *any* level-set, and still others require that it hold for *all* level-sets. We discuss these nuances when defining SXDH, but for now assume that this level-set is top-level.

Note that this is a valid extraction algorithm, since all representations of x extract to the same group element. Furthermore, this is very simple to compute (just group exponentiation and multiplication) given an encoding and a DDH tuple.

Multiplication. Multiplication of encodings given only the DDH tuple requires some more care, and the difficulty of multiplication in this case is the root cause of the complexity of our construction and proofs. To begin with, observe that given an oblique encoding of $x = (x_0, x_1, x_2, x_3)$ and a normal encoding of $y = (y_0, 0, 0, 0)$, we can exploit the homomorphic properties of the group to compute an encoding of the following product:

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

This still does not require the knowledge of the constants a, b and c and holds true since

$$e\left(g^{x_0+ax_1+bx_2+cx_3}, g^{y_0}\right) = g^{(x_0+ax_1+bx_2+cx_3)y_0} = g^{x_0y_0+ax_1y_0+bx_2y_0+cx_3y_0}.$$

On the other hand, if the encoding of y also uses the oblique representation (i.e., all slots can be used to encode the plaintext element), then evaluating the cross-product terms would require knowledge of the exponents a, b and c. This would be a problem for us, because our multiplication circuit would not work for certain encodings if we only had the DDH tuple and not the "secret" exponents. So, we have a slight dilemma: we need to enable oblique encodings for (at least) the challenge encodings so that we do not need the secret exponents a, b, and c in the clear, but we also need to ensure that we never have to multiply two oblique encodings (necessitating some kind of restriction on the oblique encodings).

Fixing Multiplication. We can avoid this issue by modifying what elements are allowed to be in oblique form. Note that all of the challenge elements, by definition, are in a (top-level) level set $\mathbf{i}$, for some binary vector $\mathbf{i}$ such that there exists an index j such that $\mathbf{i}_j = 1$ and $\mathbf{i}_k = 0$ for all $k \neq j$. If we ensured that only elements where $\mathbf{i}_j = 1$ could be encoded obliquely, then we would never have to multiply two obliquely encoded elements: multiplying two elements that have a level in common is forbidden by the definition of a multilinear map, so we could never multiply two elements for which both had $\mathbf{i}_j = 1$. Allowing oblique encodings for elements where $\mathbf{i}_j = 1$ also still allows us to obliquely encode the challenge elements, since $\mathbf{i}_j = 1$ for them by the definition of the SXDH assumption.

This, it turns out, is exactly what we do. If we go back to the definition of our encodings, we can see that our proof π_0 pertains to what form encodings take. Now we sample the language instances $z_{j',b}$ for $j' \in [n]$ and $b \in \{0, 1\}$ in a manner that ensures that, at the key step of our hybrid argument, oblique encodings are only allowed for encodings such that the level-set $\mathbf{i}$ satisfies $\mathbf{i}_j = 1$. At a high level, this would involve sampling only $z_{j,1}$ as a "yes" instance and all others as "no" instances. We can use the proof π_0 to enforce that any valid encoding must either be in normal representation, or correspond to a level-set $\mathbf{i}$ such that $\mathbf{i}_j = 1$. This constitutes the core technical idea that allows us to complete the proof. In the next section, we present more details of how to actually implement this idea.

2.3 Proof Techniques and Details

The above proof intuition explains the overall strategy for our proof. However, it doesn't explain many of the details of the construction, or how our overall proof is structured. We attempt to close some of those gaps here.

Overall Proof Structure. We start by describing the overall structure of our proof. This is not the exact structure of our hybrids (some of the steps below constitute multiple hybrids), but the overall flow is the same..

1. We start with our actual construction, with a "random" SXDH challenge tuple. The multilinear map circuits (add, invert, multiply, and extract) have access to the secret exponents a, b, and c.

2. We switch the first FHE ciphertext of our challenge encodings to oblique form. This involves a step where we "forget" the secret keys corresponding to the first FHE ciphertexts in the tuples.

3. We switch the second FHE ciphertext of our challenge encodings to oblique form. This step is almost identical to the previous step in terms of the proof.

4. We modify our circuits to enforce the special restrictions on oblique encodings described above, where oblique encodings are only allowed for terms that contain the level-set of the challenge elements.

5. Our circuits "forget" a, b, and c, and use the tuple $\left(g, g^a, g^b, g^c\right)$ instead when needed in the circuit algorithms. Note that the DDH tuple itself is a "random" tuple where $c \leftarrow \mathbb{Z}_q$ uniformly at random.

6. We switch the DDH tuple from "random" to "real" (i.e. $c = ab$). As a result, the challenge encodings, which are in oblique representation, get automatically switched from "random" to "real" SXDH encodings over our MMap.

7. We run all of the above steps except for (5) in reverse and "clean up."

The overview above accurately describes our high-level hybrid structure. In the actual proof, these high-level or "outer" hybrids are often sub-divided into sequences of "inner" hybrids to deal with the interplay between the NIZK proof system and the piO scheme. In the rest of this subsection, we will look at a few steps/hybrids of the proof in more detail. We believe that these will provide intuition for how the proofs work.

OR Proofs with Hard Languages. One of the key ingredients in our proof that we use repeatedly is what we call an "OR proof." Suppose we have some "hard" language $\mathcal{L}$ where we can efficiently sample both "yes" and "no" instances of $\mathcal{L}$. Furthermore, we require that (a) it is hard to efficiently distinguish between "yes" and "no" instances of $\mathcal{L}$, that each "yes" instance of $\mathcal{L}$ has a *unique* language membership witness, and that there is an efficient verification function of the form $R_{\mathcal{L}}$: instance $\times$ witness $\rightarrow \{0,1\}$. We define these languages formally in definition 3.4 and note that the existence of such $\mathcal{L}$ follow from the existence of DDH-hard groups.

Our OR proofs will typically be statements of the following form:

- **Either** an MMap encoding satisfies some property (e.g., consistency or normal representation).

- **Or** there is a witness wit_x for an instance x of $\mathcal{L}$ such that $R_{\mathcal{L}}\left(x, \mathsf{wit}_x\right) = 1$.

These OR proofs allow us to subtly change things in our security proofs while still allowing us to use piO (which requires input/output equivalence of programs up to "randomness"). The interplay between the language $\mathcal{L}$ and our zero knowledge proof system is key to these OR proofs' usefulness.

Recall that we require a dual-mode NIZK proof system that is perfectly sound and extractable in the binding mode, and perfectly witness indistinguishable and perfectly zero-knowledge in the hiding mode. Our proofs will typically start out in binding mode. The perfect binding property allows us to ensure that, if $x \notin \mathcal{L}$, then such a proof will never verify in binding mode unless the property check passes.

Next, suppose that we have an MMap circuit that, when handling the OR proof(s) in the input encoding(s), uses an extraction trapdoor $\mathsf{t}_{\mathsf{ext}}$ to extract a language-membership witness wit_x for x from a verifying proof of membership for x. Now consider an alternative version of the same circuit that is identical except that it uses a hardwired language-membership witness wit'_x for the same x. Since instances of $\mathcal{L}$ have unique witnesses, then we know that the extracted witness wit_x must be the same as the hardwired witness wit'_x. Therefore, the output distribution of these two programs is identical. Now, if needed in some hybrid of the proof, we could make this switch, and invoke piO security against X-IND samplers to argue that the obfuscations of the two circuit versions are computationally indistinguishable.

We also extensively use hiding mode in our OR proofs. In particular, suppose we have two versions of an MMap circuit that are identical except that they generate valid proofs with different witnesses as part of the output encodings. For example, the first version generates a verifying proof of consistency, while the second version generates a verifying membership proof for some $x \in \mathcal{L}$. Since the NIZK proof system is perfectly witness indistinguishable in the hiding mode, the output distributions of both versions of the circuit are identical. So yet again, if needed in some hybrid of the proof, we could make this switch, and invoke piO security against X-IND samplers to argue that the obfuscations of the two circuit versions are computationally indistinguishable.

Finally note that we can switch our NIZK proof systems between hiding mode and binding mode in a computationally indistinguishable manner. This follows from the fact that the public parameters of our NIZK proof system can be selected independently from (and before) the rest of the construction (circuits, etc.) is generated. We exploit this in several hybrids of our proof.

Enabling Oblique Encodings. With our OR proofs in mind, we can describe how we enable oblique encodings in our construction. Below, we describe how we enable oblique encodings for all terms (which turns out to be the first step in our overall proof). Each of these steps is rather complicated and consists of a number of sub-steps, which we outline below. We assume the challenge terms are in some level $\mathbf{i} = (\mathbf{i}_1, \ldots, \mathbf{i}_n)$ such that $\mathbf{i}_j = 1$ for some $j \in [n]$.

1. We start with the obfuscated circuits of the actual scheme, and "random" challenge encodings in normal form. All of our language instances are unsatisfiable and our proof systems are in binding mode and generate witnesses using extraction trapdoors.

2. We change the language instances $z_{j,0}$ and $z_{j,1}$ to be members. Note that this is a simple step in the proof, since we can simulate the rest of the components of the multilinear map given the language instances.

3. We modify the obfuscated MMap circuits so that they directly use the hardwired witnesses $\mathsf{wit}_{z_{j,0}}$ and $\mathsf{wit}_{z_{j,1}}$ for these instances in order to generate verifying proofs π_0 as part of their output encodings. We note that this effectively allows *all* encodings to be oblique.

4. We switch the proofs π_0 of the challenge encodings to language-membership proofs using the witness $\mathsf{wit}_{z_{j,1}}$, rather than proving normal representation using some other witness (in particular, the FHE secret keys).

While step 3 is simple, step 4 is more complicated: in order to switch witnesses, we need to switch our NIZK proof system to hiding mode. Our NIZK proof system is only guaranteed to have perfect witness indistinguishability in hiding mode, so step 4 actually consists of several steps.

Later in the proof, when we only want to enable oblique encodings for elements such that $\mathbf{i}_j = 1$, we will use a similar argument, but only modify the proofs and witnesses for elements where $\mathbf{i}_j = 1$.

Indistinguishability of Encoding Representations. One of the core steps in our proof is switching the FHE ciphertexts (one at a time) from encodings in normal form to encodings in oblique form (so that we can eventually embed a DDH challenge). Each of these steps is rather complicated and consists of a number of sub-steps, which we outline below. We only describe the first FHE ciphertext switch below, as the second is almost identical to the first. As before, we assume the challenge terms are in some level $\mathbf{i} = (\mathbf{i}_1, \ldots, \mathbf{i}_n)$ such that $\mathbf{i}_j = 1$ for some $j \in [n]$.

1. We start by assuming that our circuits use the appropriate hardwired witnesses to output valid oblique encodings, as discussed above. The FHE ciphertexts in the challenge encodings themselves are still in the normal form. However, the proofs of normal encoding have been replaced with proofs of language membership, as discussed earlier.

2. We next change the language instance y (used in π_1) to be a satisfying or "yes" instance in $\mathcal{L}$.

3. Our circuits "forget" the secret key for the first FHE ciphertext in the encodings. We evaluate addition and multiplication homomorphically on the first FHE ciphertext, which allows us to create new encodings that are correctly distributed. In the extract circuit, we just use the second FHE ciphertext (for which we still have the secret keys) and ignore the first FHE ciphertext. At a high level, we justify these steps by invoking piO security and the perfect soundness of the NIZK proof system in the binding mode).

4. We modify the MMap circuits so that they always use the witness wit_y to generate proofs for π_1 rather than proving the consistency check natively (i.e. by using the secret keys of the FHE scheme). This is a crucial step that requires multiple hybrids and involves many alterations to the obfuscated MMap circuits (all of which are justified by invoking piO security and the perfect witness indistinguishability of the NIZK proof system in hiding mode).

5. We switch the proofs π_1 of the challenge encodings to language-membership proofs using the witness wit_y, rather than proving consistency using the FHE secret keys. In this step, we again crucially rely on the perfect witness indistinguishability of the NIZK proof system in the hiding mode.

6. At this point, all of the MMap circuits and the challenge encodings have essentially "forgotten" the secret key for the first FHE ciphertext. So we can switch the first FHE ciphertext in each of the challenge tuples from normal form to the special oblique form described earlier.

7. We repeat all of the above steps except for step 6 in the reverse order.

There are several points in the above outline that merit explanation. To begin with, forgetting the first FHE secret key in the MMap circuits (step 3) is tricky. The need to compute additions inversions and multiplications of encodings without secret keys is why we need to use FHE rather than a more basic form of encryption. We first note that the output distributions of the addition, inversion, multiplication, and extraction circuits are correct. However, proving this turns out to have some subtle difficulties.

By the definition of FHE, adding and multiplying two FHE ciphertexts results in a valid FHE ciphertext. But in order to apply piO, we actually need the results of homomorphically adding or multiplying two FHE ciphertexts to be identically distributed to a fresh ciphertext–otherwise the output distributions are not the same. In addition, we need decryption to always work. This means that our FHE scheme must have perfect correctness and a property we call "well-distributedness of output of $\mathsf{FHE.Eval}$" (called "compactness" in [FHHL18]). While LWE-based FHE schemes do not have these properties, piO-based FHE schemes such as the one in [CLTV15] do satisfy these requirements, so we can assume that FHE with these properties exists. This allows us to justify that the output distributions of the addition and multiplication circuits remain unaltered. The justification for the inversion circuit is essentially identical.

Extraction is simpler: we can just ignore the first FHE ciphertext in each tuple with regards to extraction since we can learn all of the information we need to compute the extraction value from the second FHE ciphertext. We rely on the perfect soundness guarantees of the NIZK proof system in the binding mode to argue that an adversary cannot fool the extraction circuit into accepting encodings that have inconsistent encodings. So the output distribution of the extraction circuit also remains unchanged.

Next, step 4 entails that the MMap circuits be suitably modified so that they no longer check the input encodings for consistency when deciding which witness to use for the output proof π_1. Since we are potentially switching proof witnesses here, we crucially rely on the perfect witness indistinguishability of the NIZK proof system in the hiding mode. However, our proof hybrids are designed in a manner that the adversary cannot exploit this relaxation to design valid encodings where the FHE ciphertexts do not encode the same element. In particular, we rely on the indistinguishability of the NIZK system in the binding and hiding modes for this guarantee to hold.

Finally, we note that at the conclusion of step 5, a simulator can simulate the obfuscated MMap circuits and the challenge encodings without any information about the secret key corresponding to the first FHE encryption in each encoding. This allows us to switch the first FHE ciphertext in each challenge encoding from normal to oblique form in step 6.

Forgetting the Secret Exponents. Once the challenge encodings have been switched from normal to oblique representation entirely, we require yet another "forgetting" step - one which essentially forms the core of our proof of SXDH-hardness. We need the MMap circuits to "forget" the secret exponents a, b and c sampled at setup and used for consistency checks/extraction over oblique encodings. More specifically, we wish to switch to MMap circuits that are only hardwired with the tuple $\left(g, g^a, g^b, g^c\right)$ into our circuits, albeit without changing any functionality. This is a rather involved step in our proof since it involves applying piO to some carefully constructed circuits.

We explain the circuit switches and how to apply piO to them in greater detail. As explained earlier, the addition and inversion circuits do not require the knowledge of the secret exponents to add/invert encodings in oblique representation. Similarly, the extraction circuit can compute the extraction output on an oblique encoding $x = (x_0, x_1, x_2, x_3)$ as:

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

which is efficiently computable given only the tuple $\left(g, g^a, g^b, g^c\right)$. So the key focus here is our MMap multiplication circuit.

We can describe our MMap multiplication circuit in a very particular way: we use the following "if tree" to handle multiplication between encodings:

- **If** both FHE ciphertexts are in normal form, multiply using the trivial algorithm.
- **Else If** the first FHE ciphertext is in normal form and the second is in oblique form, use the oblique multiplication algorithm (that doesn't need the values of a, b, and c).
- **Else If** the FHE second ciphertext is in normal form and the first is in oblique form, use the oblique multiplication algorithm with the order of ciphertexts reversed.
- **Else** use the multiplication algorithm that uses the exponents a, b, and c in the clear.

At a high level, what does this convoluted representation of our multiplication circuit buy us? Suppose that in some hybrid, we force all the NIZK proofs in any valid encoding to be binding. Additionally, we fix $j^* \in [n]$ and sample the language instances $z_{j,b}$ for $j \in [n]$ and $b \in \{0, 1\}$ such that only $z_{j^*,1}$ is an "yes" instance and every other instance is a "no" instance. This would enforce that any valid encoding must either be in the normal representation or must correspond to a level $\mathbf{i} = (\mathbf{i}_1, \ldots, \mathbf{i}_n)$ such that $\mathbf{i}_{j^*} = 1$. In other words, any two valid encodings in oblique representation must be incompatible for multiplication.

It is easy to see that in such a scenario, the final **Else** branch of the MMap multiplication circuit–the only part of the circuit that needs to use the exponents a, b, and c in the clear–is never satisfied. Due to the perfect soundness of our NIZK proofs in the binding mode, no adversary (even computationally unbounded) can craft an input to our multiplication circuit that satisfies this branch. So, if we create a modified MMap multiplication circuit that is identical to the real circuit except that it does not contain the last **Else** branch, the outputs remain unchanged. Hence, the piO obfuscations of these circuits are computationally indistinguishable. This allows us to "forget" the secret exponents a, b, and c from our MMap circuits, and allows us hardwire them with a DDH challenge instance.

Of course, the above description is a simplification: our actual multiplication circuit is more complicated as some components of the output encoding need to be computed homomorphically, and we have omitted dealing with the NIZK proofs in the description above. But the intuition behind forgetting the secret exponents is exactly as described above.

2.4 Other Graded Encodings

So far, we have focused on our construction of an SXDH-hard asymmetric MMap. However, it turns out that understanding the SXDH-secure construction is almost sufficient for understanding all of the other constructions: our other constructions of asymmetric MMaps (where the joint-SXDH and the exponent-DDH assumptions hold) fit in the same overall proof framework.

In particular, in all of these proofs, we start with a "basic" group assumption (i.e. over a group without any kind of pairing), embed it in our challenge encodings, and show that any adversary that breaks the multilinear map assumption can be used to break the "basic" group assumption. 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) as is the case with the SXDH and joint-SXDH assumptions. 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. This causes some difficulty in our proofs: we need to be able to multiply challenge encodings with each other, which is not something that our proof structure can handle. At a high level, we get around this issue by strengthening the hardness assumption on the basic group $\mathbb{G}$ used in the construction.

An example of this is the following: 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, in this case an adversary can compute all encodings of the form $\mathsf{Encode}\left(\alpha^{i+(\ell+1)j}\right)$ for all $i+j<\ell$. If we wanted to put these challenge terms "in the exponent" of a basic group like in our asymmetric constructions, we would need to provide all terms of the form $g^{\alpha^{i+(\ell+1)j}}$ for all $i+j<\ell$ for some $g \leftarrow \mathbb{G}$. We can obviously not simulate this in a "basic" group given only $\left(g^{\alpha}, g^{\alpha^{\ell+1}}\right)$.

So 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:

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

where $g \leftarrow \mathbb{G}$ and $\alpha, \alpha^* \leftarrow \mathbb{Z}_q$. Note that both sets of these terms are consistent with what an adversary could check using the multilinear map. 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 $\mathbb{G}$.

Finally, it is worth noting that the $(\ell+1)$-EDDH assumption over a degree-ℓ symmetric multilinear map implies many of the most commonly studied and used assumptions over symmetric multilinear maps. For a full discussion and reductions between all of the assumptions, we refer the reader to [EHK+13].

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

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

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

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

For our constructions (similar to [FHHL18]), we need dual-mode NIZK proof systems that are

- perfectly complete in both modes;
- perfectly extractable and perfectly sound in the binding mode;
- perfectly zero knowledge and perfectly witness indistinguishable in the hiding mode.

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 $m_1, \dots, m_\ell$ if $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$ and $c_i \leftarrow \mathsf{Enc}(\mathsf{pk}, m_i)$ then
$$\Pr[\mathsf{Dec}(\mathsf{sk}, \mathsf{Eval}(\mathsf{pk}, c_1, \dots c_\ell)) = f(m_1, \dots, m_\ell)] = 1.$$
- 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 remark that for our constructions we need an FHE scheme with the following two properties:

- Perfect correctness: For any message m in the message space $\mathcal{M}$, it holds that
$$\Pr[\mathsf{Dec}(\mathsf{sk}(\mathsf{Enc}(\mathsf{pk}, m))) = m] = 1,$$
where $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$.
- Well-distributedness of output of Eval algorithm: For any (boolean) function $f : \mathcal{M}^\ell \to \mathcal{M}$ and any sequence of ℓ messages $m_1, \dots, m_\ell$ if $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$ and $c_i \leftarrow \mathsf{Enc}(\mathsf{pk}, m_i)$, it holds that
$$\mathsf{Enc}(\mathsf{pk}, f(m_1, \dots, m_\ell)) \equiv \mathsf{Eval}(\mathsf{Enc}(\mathsf{pk}, m_1), \dots, \mathsf{Enc}(\mathsf{pk}, m_\ell))$$
where $\equiv$ means that the two distributions are identically distributed and each encryption uses a fresh and independent randomness.

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:

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

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 $\text{poly}(|C|, \lambda)$.

- 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 $\left|\Pr[\mathcal{A}_{\mathsf{Exp}_0} = 1] - \Pr[\mathcal{A}_{\mathsf{Exp}_1} = 1]\right| \le \text{negl}$.
  - For every PPT attacker $\mathcal{A}$ and every sampler $S \in \mathcal{S}$, if $(C_0, C_1, \mathsf{str}) \leftarrow S$ we have

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

We also recall the definition of X-Ind samplers from [CLTV15].

Definition 3.7. The class $\mathcal{S}^{X-\mathsf{Ind}}$ of (static-input) X-Ind samplers for a circuit family $\mathcal{C}$ contains all circuit samplers $D = \{D_\lambda\}$ for $\mathcal{C}$ with the following property: For every λ, there is a set $\mathcal{X} = \mathcal{X}_\lambda$ of size at most $X(\lambda) \le 2^\lambda$ such that

- With overwhelming probability over the choice of $(C_0, C_1, z) \leftarrow D_\lambda$, for every $x' \notin \mathcal{X}$, it holds that $C_0(x', r) = C_1(x', r)$ for every random string r.

- For every PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, the advantage of $\mathcal{A}$ in the following experiments is $\text{negl} \cdot X^{-1}$:

1. $(x, \mathsf{st}) \leftarrow \mathcal{A}_1(1^\lambda)$.
2. $(C_0, C_1, z) \leftarrow D$.
3. $y \leftarrow C_b(x)$ where $b \leftarrow \{0,1\}$.
4. $b' \leftarrow \mathcal{A}(\mathsf{st}, C_0, C_1, z, x, y)$.

The advantage of $\mathcal{A}$ is defined to be $\Pr[b = b'] - 1/2$.

Definition 3.8. Let X be a by function such that $X \leq 2^\lambda$. We say that a uniform PPT algorithm X-piO is an X-piO for randomized circuits if it is a piO for the class of X-Ind samplers $\mathcal{S}^{X\text{-}\mathsf{Ind}}$ over $\mathcal{C}$ that includes all randomized circuits of size at most λ.

4 Multilinear Maps

In this section, we define multilinear maps (graded encodings) and some of their properties. While most of this section is standard and thus not new to a reader familiar with the area, we do emphasize that we have some new material. In particular, we need to delve into the security games of asymmetric multilinear maps and related assumptions such as SXDH and joint-SXDH in ways that, to our knowledge, have not been previously explored.

We start by defining some basic notation around multilinear maps. Traditionally, a (symmetric) multilinear map was defined as a deterministic function $e : \mathbb{G}^n \to \mathbb{G}_T$ where plaintexts α were encoded as g^α for some generator $g \in \mathbb{G}$, with the property that $e\left(g^{\alpha_1}, \ldots, g^{\alpha_n}\right) = e\left(g, \ldots g\right)^{\prod_{i=1}^n \alpha_i}$. However, unfortunately we cannot construct such nice and simple multilinear maps, as our constructions will have randomized encodings. Moreover, we will build multilinear maps with the additional property that such maps can be partially computed, or "graded." Technically speaking, we will build graded encodings, but we will follow the literature and use these terms interchangeably.

We refer to elements of multilinear maps as *encodings*, which encode some value. Each encoding is present at a given *level* of the multilinear map, which, informally speaking, refers to how many multiplications (and, in the asymmetric case, with which elements) it takes to reach the encoding from "top-level" encodings. We explain this below.

Level-i Encoding, Symmetric Case. For a degree n multilinear map, we define a level-0 encoding to be a plaintext element. We define a level-1 encoding to be an encoding of some element in the source group. For $1 \leq i \leq n$, we define a level-i encoding to be the product of i level-1 encodings, so that a level-n encoding is an encoding in the target group.

Symmetric Multilinear Map. We are now ready to define a symmetric multilinear map. Our definitions are inspired by [GGH13a]. Let R be a ring and let 0_R be the zero element for the ring.

Definition 4.1. (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 or the encodings are not at the same level).
- $\mathsf{Inv}(\mathsf{pp}, [a]_i)$: 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 or $i_1 + i_2 > n$).
- $\mathsf{Ext}(\mathsf{pp}, [a]_i)$: 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). For any two valid encodings that encode the same plaintext element $a \in R$ at the same level i, we require that the output of Ext be identical.
- $\mathsf{ZeroTest}(\mathsf{pp}, [a]_i)$: 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 4.2. Note that equality-checking of encodings at any valid level follows implicitly from the addition, inversion and zero-test operations at the same level, so for some applications we may not need the extract algorithm. However, we can achieve it in our constructions so we provide it here for completeness.

We can now move on to our definition of an asymmetric multilinear map. Traditionally, asymmetric multilinear maps were defined in a similar way as symmetric multilinear maps–as functions $e : \mathbb{G}_1 \times \ldots \times \mathbb{G}_n \to \mathbb{G}_T$ where plaintexts α were encoded as g_i^α for some generators $g_i \in \mathbb{G}_i$, with the property that $e\left(g_1^{\alpha_1}, \ldots, g_n^{\alpha_N}\right) = e\left(g_1, \ldots g_n\right)^{\prod_{i=1}^n \alpha_i}$. As with symmetric multilinear maps, we will not construct or define this kind of map, but instead build (randomized) graded encodings.

Level-i Encoding, Asymmetric Case. Denoting the level of an encoding in an asymmetric multilinear map is more complicated than in the symmetric case. We cannot simply use an integer to indicate the level of an encoding because we need to keep track of which "groups" have been multiplied in order to generate the encoding.

To do this, we define a "level-vector" $\mathbf{i} \in \{0,1\}^n$. Informally, speaking, the level-vector $\mathbf{i}$ has a one in position j if and only if the encoding corresponding to the level set vector corresponds to some product that included an element from the jth "level one" encoding set (i.e. $\mathbb{G}_j$), where the "level-one" encoding corresponds to a level-vector with a single non-zero entry.

We denote $\vec{0}_N$ and $\vec{1}_n$ to be be the all-zeroes and all-ones vectors of length n, respectively. With this in mind, we can define a level-$\vec{0}_n$ encoding to be a plaintext element as we did before. A level-$\vec{1}_n$ encoding is analagous to an encoding of some element in the "source group."

For two level-vectors $\mathbf{i}_1, \mathbf{i}_2 \in \{0,1\}^n$, we say that $\mathbf{i}_1 \leq \mathbf{i}_2$ if every entry of $\mathbf{i}_2$ is smaller than its corresponding entry (coordinate-wise) in $\mathbf{i}_1$. A level-vector $\mathbf{i} \in \{0,1\}^n$ is said to represent a valid level if $\vec{0}_n \leq \mathbf{i} \leq \vec{1}_n$. We say two level-vectors $\mathbf{i}_1$ and $\mathbf{i}_2$ are *pairing-compatible* if $\mathbf{i}_1 + \mathbf{i}_2 \leq \vec{1}_n$. We note that our definition of level-sets implicitly enforces the closure restriction of pairing-compatibility as discussed in [LV16], so we do not need to worry about closure of pairing-compatibility here.

Asymmetric MMap. We can now define an asymmetric multilinear map. Let R be a ring and let 0_R be the zero element for the ring.

Definition 4.3. (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 or the level-vectors are not identical).
- $\mathsf{Inv}(\mathsf{pp}, [a]_{\mathbf{i}})$: 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 $\mathbf{i}_1 + \mathbf{i}_2 \leq \vec{1}_N$, 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 or $\mathbf{i}_1 + \mathbf{i}_2 \nleq \vec{1}_n$).
- $\mathsf{Ext}(\mathsf{pp}, [a]_{\mathbf{i}})$: 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). For any two valid encodings that encode the same plaintext element $a \in R$ with the same level-vector $\mathbf{i}$, we require that the output of Ext be identical.
- $\mathsf{ZeroTest}(\mathsf{pp}, [a]_{\mathbf{i}})$: 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.

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.

4.1 Symmetric Multilinear Map Assumptions

We begin by defining the main hardness assumption over symmetric multilinear maps that we use in this paper.

The $(n+1)$-EDDH Assumption. We define the $(n+1)$-exponential DDH (EDDH) assumption over symmetric multilinear maps. Suppose that we have a multilinear map of degree n. Informally, the $(n+1)$-EDDH assumption states that an encoding of some random element and an encoding of that element raised to the $n+1$th power are indistinguishable from two encodings of random elements. Note that this is plausible since, for some encoding $[\alpha]_1$, we can only hope to compute $[\alpha^n]_1$ using the multilinear map.

As discussed in [EHK+13], we note that the $(n+1)$-EDDH assumption implies many of the most commonly studied assumptions over (symmetric) multilinear maps. We do not define them all here or go into detail about these reductions, but instead encourage the reader to refer to that paper. We formally define the $(n+1)$-EDDH assumption below.

Definition 4.4. (**$(n+1)$-EDDH Assumption.**) The $(n+1)$-EDDH assumption over a degree-n symmetric multilinear map with input ring R requires that the following computational indistinguishability holds:

$$\left([\alpha]_1, [\alpha^{n+1}]_1\right) \quad \overset{c}{\approx} \quad \left([\alpha]_1, [\alpha^*]_1\right),$$

where $\alpha, \alpha^* \leftarrow R$, and $[\alpha]_1$, $[\alpha^{n+1}]_1$ and $[\alpha^*]_1$ are all level-1 encodings.

4.2 Asymmetric Multilinear Map Assumptions

We next define our asymmetric multilinear map assumptions. Unfortunately, defining assumptions on asymmetric multilinear maps is much more complicated than doing so for symmetric multilinear maps. We will explain the details in this section.

k-EDDH Assumption. We start by presenting the k-EDDH assumption for asymmetric multilinear maps. Like the $(n+1)$-EDDH assumption we defined earlier for symmetric assumption, the k-EDDH assumption for asymmetric maps does involve giving out an encoding of some element α as well as an encoding of α^k. However, in the asymmetric case, we require that they be at the same level of the multilinear map. This extra restriction allows the assumption to (potentially) be valid for any $k \geq 2$, rather than only viable for $k > n$.

Definition 4.5. (**k-EDDH Assumption.**) Consider a degree-n asymmetric multilinear map over some ring R. Let $\alpha_i, \alpha_i^* \leftarrow R$ be sampled uniformly at random for $i \in [1, n]$. In addition, let $\mathbf{u}_i \in \{0,1\}^n$ denote the vector with i^{th} coordinate one and all other coordinates zero.

The k-EDDH assumption over a degree-n multilinear map requires that the following computational indistinguishability holds:

$$\Big([\alpha_1]_{\mathbf{u}_1}, [\alpha_1^k]_{\mathbf{u}_1}\Big), \ldots, \Big([\alpha_n]_{\mathbf{u}_n}, [\alpha_n^k]_{\mathbf{u}_n}\Big) \overset{c}{\approx} \quad \Big([\alpha_1]_{\mathbf{u}_1}, [\alpha_1^*]_{\mathbf{u}_1}\Big), \ldots, \Big([\alpha_n]_{\mathbf{u}_n}, [\alpha_n^*]_{\mathbf{u}_n}\Big).$$

Note that in the above definition we give out random, independent challenge encodings in each of the n top-level vector-sets (i.e. source groups). This allows us to generate an encoding of α and α^k in *any* level-vector using multiplication by the identity element encoded at different levels. The definition is similar in spirit to the definition of SXDH in [Lin17].

SXDH Assumption. Intuitively, the SXDH assumption says that, on a multilinear map, "DDH" is hard in every "source group" [BS03, Rot13]. We generalize this definition to graded encodings and formalize it definition below.

Definition 4.6. (**SXDH Assumption.**) Consider a degree-n asymmetric multilinear map over some ring R. Let $\alpha_{i,0}, \alpha_{i,1}, \alpha_{i,2} \leftarrow R$ be sampled uniformly at random for $i \in [1, n]$. In addition, let $\mathbf{u}_i \in \{0,1\}^n$ denote the vector with ith coordinate one and all other coordinates zero.

The SXDH assumption over a degree-n multilinear map requires that the following indistinguishability holds:

$$\Big([\alpha_{1,0}]_{\mathbf{u}_1}, [\alpha_{1,1}]_{\mathbf{u}_1}, [\alpha_{1,0} \cdot \alpha_{1,1}]_{\mathbf{u}_1}\Big), \ldots, \Big([\alpha_{n,0}]_{\mathbf{u}_n}, [\alpha_{n,1}]_{\mathbf{u}_n}, [\alpha_{n,0} \cdot \alpha_{n,1}]_{\mathbf{u}_n}\Big)$$

$$\overset{c}{\approx} \quad \Big([\alpha_{1,0}]_{\mathbf{u}_1}, [\alpha_{1,1}]_{\mathbf{u}_1}, [\alpha_{1,2}]_{\mathbf{u}_1}\Big), \ldots, \Big([\alpha_{n,0}]_{\mathbf{u}_n}, [\alpha_{n,1}]_{\mathbf{u}_n}, [\alpha_{n,2}]_{\mathbf{u}_n}\Big).$$

Note that in the above definition we give out random, independent DDH-style challenge encodings for each of the n top-level vector-sets (i.e. source groups). This is equivalent to the definition in [Lin17], although we explicitly provide challenges in every top level and they do not.

We note that some papers on graded encodings are either unclear about specifically on which vector-sets "DDH" must hold or define SXDH such that "DDH" must hold on *every* vector set [LV16]. We call this assumption *uber-SXDH* and define it below. We mainly include this for completeness, and for comparison with the joint-SXDH assumption (which we define below).

Definition 4.7. Uber-SXDH Assumption The uber SXDH assumption over a degree-n asymmetric MMap requires that the following indistinguishability holds for all valid level-vectors $\mathbf{i}$ such that $\mathbf{0}_n < \mathbf{i} \leq \vec{1}_n$:

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

where $\alpha_0, \alpha_1, \alpha^* \leftarrow R$.

This definition may seem identical to regular SXDH, and it is reducible using a simple hybrid argument to and from our SXDH definition for constant-degree multilinear maps. However, for large-degree multilinear maps, the assumptions are not equivalent: the Uber-SXDH assumption becomes an exponential family of assumptions (one for every possible level-vector, for a total of 2^n).

To see why this is the case, consider the following black-box adversary $\mathcal{A}$. Suppose $\mathcal{A}$ has some randomly selected level-vector $\mathbf{v}$ hard-coded inside it. When given an SXDH challenge tuple, $\mathcal{A}$ first checks if the level-vector is equal to $\mathbf{v}$. If it is, then it somehow magically can tell whether the tuple is random or not. If not, then it aborts and outputs $\perp$. Since $\mathcal{A}$ is a black box, we cannot learn the value of $\mathbf{v}$ except by randomly querying. So $\mathcal{A}$ cannot be used to break the regular SXDH game efficiently if n is polynomial in λ, because it will be impossible to find $\mathbf{v}$ efficiently by random guessing, which is the best we can do with our black-box adversary $\mathcal{A}$. However, the existence of $\mathcal{A}$ clearly breaks the Uber-SXDH assumption, because there is some level-vector–$\mathbf{v}$–where an adversary can distinguish between the two classes of tuple.

For most practical purposes, though, the two assumptions are interchangeable: the "proper" SXDH assumption implies that "DDH" is hard in *any* level-set, even if the adversary can decide on a challenge level-vector after seeing the public parameters of the multilinear map. This follows from the fact that every possible level-set can be derived through multiplication from one of the SXDH challenge level-vectors $\mathbf{u}_1, \ldots, \mathbf{u}_n$. In addition, a simple hybrid argument can extend this to any polynomial number of adversarially-chosen level-sets, which is good enough for polynomial-time reductions. However, a proper reduction from *any* level-set to *all* level-sets would still take time proportional to the number of level sets, so we unfortunately cannot say that these assumptions are equivalent.

We further note that our definitions are similar to [Lin17]) and borrow from the bilinear SXDH definition of [ABBC10], which also outputs a DDH challenge in each source group. We note that the Uber-SXDH assumption is more similar in flavor to the definitions from [CLL$^+$13], although they are of course equivalent for bilinear maps. Finally, we encourage the reader to peruse [BMZ19] for an interesting discussion on how minute changes in definitions can make substantial differences.

Joint-SXDH Assumption. The joint-SXDH assumption can be seen as a generalization of the Uber-SXDH assumption. Informally, instead of requiring "DDH" to just hold on all level-sets, we require it to hold simultaneously on all possible sets of non-pairable level-sets. We give out "DDH" tuples of elements *with the same secrets* corresponding to many different level-vectors, with the caveat that the level-vectors cannot be multiplied with each other. We formalize this in the definition below. Our definition mirrors that of [LV16], where the assumption was first made.

Definition 4.8. Joint-SXDH Assumption. Let T be some set of size $s = |T| = \text{poly}(\lambda)$, 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.

The s-joint-SXDH assumption over a degree-n asymmetric MMap over some ring R requires that the following indistinguishability holds for every set T of size s:

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

where $\alpha_0, \alpha_1, \alpha^* \leftarrow R$ uniformly at random.

On Adaptivity and Exponential-Sized Assumption Families. Unlike SXDH, the j-SXDH assumption must hold on *every* set T of size s where no members of T are pairing-compatible. There are 2^{2^n} sets of possible level-vectors, and while many of these sets contain pairing-compatible vectors, some do not. As an example, any set where all of the level-vectors have one as their first entry has no pairing-compatible level-vectors, and there are $2^{2^{n-1}}$ of these sets. This means that proving j-SXDH for a multilinear map with large n will be very difficult.

However, in practice we usually do not need our multilinear map assumptions to hold over "all" level-sets simultaneously. In reality, in any protocol (or any step in a security game), it is more likely that the adversary can adaptively choose a particular set of level-vectors to attack, upon which they will repeatedly receive a tuple of challenge elements at that particular level-vector. Hybrid arguments involving multilinear maps might repeat this process a polynomial number of times, but most security proofs that do not have exponential loss will go through with a polynomial number of such challenges. Few security proofs need *every* possible set to be secure, and those that do typically already use reductions that involve exponentially secure primitives.

We can model the intuition behind this alternative definition of the j-SXDH assumption in a game-based way, which would proceed as follows: the adversary would see the public parameters of the multilinear map, perform some polynomially bounded computation, and then submit a level-vector to the challenger. The challenger would provide either a "random" or "DDH" set of encodings to the adversary. The adversary could continue to query the challenger on level-vectors, and appropriate challenge tuples would be provided to the adversary by the challenger as long as they were not pairing-compatible with any previous tuples. At the end of the game, the adversary would have to decide which kind of encodings they received. This game gives most security proofs the power they need while also avoiding doubly exponential amounts of sets. We define it in the game below.

Game-Based Definitions for j-SXDH. We can redefine the j-SXDH assumption as a game between a challenger and an adversary, in accordance with our earlier discussion. Note that this new definition is *not* equivalent to the standard j-SXDH definition, but may be functionally equivalent in most cases.

Definition 4.9. Game-Based j-SXDH Definition. We define the game-based s-j-SXDH assumption as a game between a challenger and an adversary involving an asymmetric multilinear map of degree n over a ring R, which proceeds as follows.

- The challenger runs $\mathsf{pp} \leftarrow \mathsf{Setup}\left(1^\lambda, 1^n\right)$ and sends pp to the adversary.
- The challenger flips a coin to determine a bit $b \leftarrow \{0,1\}$ uniformly at random.
- If $b = 0$, the challenger selects $a_1, a_2, a^* \leftarrow R$ uniformly at random.
- If $b = 1$, the challenger selects $a_1, a_2 \leftarrow R$ uniformly at random and sets $a^* = a_1 a_2$.
- For $(k = 1; k \le s; k++)$:
  - The adversary can perform any $poly\left(\lambda\right)$ calculations using pp. The adversary then sends a level-vector $\mathbf{i}$ to the challenger.
  - The challenger checks if $\mathbf{i}$ is pairing-compatible with previous submissions. If so, it aborts.
  - The challenger sends the tuple of elements

    $$\left\{\mathsf{Encode}\left(pp, a_1, \mathbf{i}\right), \mathsf{Encode}\left(pp, a_2, \mathbf{i}\right), \mathsf{Encode}\left(pp, a^*, \mathbf{i}\right)\right\}$$

    to the adversary.
- The adversary computes a bit b'.

We say that the adversary wins the game if $b' = b$.

In some applications, we could relax this (adaptive) game-based definition further to a non-adaptive case, where the adversary would have to declare the set T of level-vectors it wants to attack before it sees the public parameters of the multilinear map. This could be the case for practical applications, for instance, if some specific level-vector is used in a special way in some specific cryptosystem. Of course, there may also be applications where the adversary is allowed to choose the set of level-vectors after it sees the public parameters, but with some restrictions. A good analogy is selective versus adaptive security in identity-based encryption (as discussed in [BB04], for instance). This fine point over definitions has seemingly not come up yet in the literature because, to our knowledge, there have been no attempts to formally prove complicated asymmetric assumptions on multilinear maps.

In our proofs for multilinear maps, we will need to embed challenge elements in certain levels. In other words, we need to commit to the level-vector of the challenge elements while generating the public parameters (before they are given to the adversary).[8] So our multilinear map constructions (without any other arguments) will only be non-adaptive, which is less than ideal. This may seem problematic. However, our constructions do have an additional property: the level-set of the challenge element is statistically hidden from the adversary in the real scheme, and computationally hidden from the adversary in all of our hybrids in our security proof. Therefore, we can achieve adaptive security (with some loss) by guessing the level-vector (or guessing some level-vector that allows us to generate a challenge encoding in the level-set the adversary actually chooses) in advance and hoping that the adversary chooses the same level-vector. The adversary cannot tell where we have embedded the challenges, and so thus must proceed as in an honest evaluation. We lose a factor in the security reduction proportional to the rate at which we guess correctly.

So, we can also define a nonadaptive version of the j-SXDH assumption game. This will be useful for our proofs, because as we said above, our challenges will need to be embedded in the public parameters. In this game, we allow the adversary to make the public parameters dependent on the challenge level-vectors (as would be the case in a security reduction). However, we also require that the public parameters in this case are indistinguishable from honestly generated public parameters.

Definition 4.10. Non-Adaptive Game-Based j-SXDH Definition. We define the non-adaptive game-based t-j-SXDH assumption as two games between a challenger and an adversary involving an asymmetric multilinear map of degree n over a ring R, which proceed as follows. Important changes from the game definition above are highlighted in red.

Game 1:

- The adversary selects a set T of s level-vectors and sends them to the challenger.
- The challenger verifies that the vectors in T are not pairing compatible. Otherwise, the challenger aborts.
- The challenger flips a coin to determine a bit $b \leftarrow \{0,1\}$ uniformly at random.
- If $b = 0$, the challenger selects $a_1, a_2, a^* \leftarrow R$ uniformly at random.
- If $b = 1$, the challenger selects $a_1, a_2 \leftarrow R$ uniformly at random and sets $a^* = a_1 a_2$.
- The challenger runs $\mathsf{pp} \leftarrow \mathsf{Setup}\left(1^\lambda, 1^n, T\right)$ and sends pp to the adversary.
- For $(k = 1; k \leq s; k++)$:
  - The challenger sends the tuple of elements
$$\left\{\mathsf{Encode}\left(pp, a_1, \mathbf{i}\right), \mathsf{Encode}\left(pp, a_2, \mathbf{i}\right), \mathsf{Encode}\left(pp, a^*, \mathbf{i}\right)\right\}$$
    to the adversary.
- The adversary computes a bit b'.

We say that the adversary wins game 1 if $b' = b$.

Game 2:

- The adversary selects a set T which contains up to s level-vectors and sends them to the challenger.
- The challenger verifies that the vectors in T are not pairing compatible. Otherwise, the challenger aborts.
- The challenger flips a coin to determine a bit $b \leftarrow \{0,1\}$ uniformly at random.
- If $b = 0$, the challenger runs $\mathsf{pp} \leftarrow \mathsf{Setup}\left(1^\lambda, 1^n, T_0\right)$
- If $b = 1$, the challenger runs $\mathsf{pp} \leftarrow \mathsf{Setup}\left(1^\lambda, 1^n\right)$
- The challenger sends pp to the adversary.

[8] This seems to us to be something that is necessary for rigorous proofs of multilinear maps, but we cannot be sure this is this case nor come up with a convincing argument.

- The adversary computes a bit b'.

We say that the adversary wins game 2 if $b' = b$.

We say that the *advantage* of an adversary in breaking the non-adaptive game-based t-j-SXDH assumption is the maximum of its advantage in games 1 and 2.

It may seem excessive to define the non-adaptive version of the game in the way that we did. But we need the "public parameter indistinguishability" property of game 2 to avoid some pathological schemes and for proofs to go through. In fact, we can prove the following lemma relating the two security games.

Lemma 4.11. *Consider some asymmetric multilinear map of degree n over a ring R. Any adversary $\mathcal{A}$ that can win the game-based j-SXDH game as in definition 4.9 with advantage ϵ can be used to win the non-adaptive game-based j-SXDH game as defined in definition 4.10 with advantage $\frac{\epsilon}{2^{nt+1}}$.*

Proof. Suppose we are given an adversary $\mathcal{A}$ that can win the game-based j-SXDH game with advantage ϵ. We generate a new adversary $\mathcal{A}'$ that has advantage $\frac{\epsilon}{2^{nt+1}}$ in the non-adaptive game-based j-SXDH game in the following way.

First, suppose that $\mathcal{A}$ cannot win Game 2 with advantage $\frac{\epsilon}{2^{nt+1}}$. Otherwise, we are done. Assuming that $\mathcal{A}$ cannot win game 2, we create a new adversary for game 1 called $\mathcal{A}'$ in the following way:

- $\mathcal{A}'$ samples a random set T of s level-vectors.
- $\mathcal{A}'$ sends T to the challenger.
- $\mathcal{A}'$ receives pp from the challenger and forwards it to $\mathcal{A}$.
- $\mathcal{A}'$ also receives, for all $i \in [1, s]$ encodings of the form

$$\{\mathsf{Encode}(pp, a_1, \mathbf{i}), \mathsf{Encode}(pp, a_2, \mathbf{i}), \mathsf{Encode}(pp, a^*, \mathbf{i})\}$$

  from the challenger.
- For $(k = 1; k \leq t; k++)$:
  - $\mathcal{A}$ outputs a level-vector $\mathbf{v}$ as a query. If $\mathbf{v} \in T$, then $\mathcal{A}'$ forwards the appropriate encodings to $\mathcal{A}$. Otherwise $\mathcal{A}$ aborts.
- Eventually $\mathcal{A}$ outputs some bit b'. We let $\mathcal{A}'$ also output b'.

If we assume that the public parameters pp were generated honestly, then $\mathcal{A}'$ outputs some bit with probability $\frac{1}{2^{nt}}$. In this case, a transcript of the interaction between $\mathcal{A}$ and $\mathcal{A}'$ would be itentical to that of $\mathcal{A}'$ and a challenger in the adaptive game. Since $\mathcal{A}$ has advantage $\leq \frac{\epsilon}{2^{nt+1}}$ in game 2, by a union bound it must have advantage at least $\frac{\epsilon}{2^{nt+1}}$ in game 1, completing the proof. □

5 An Asymmetric MMap Construction

In this section, we show a how to construct an asymmetric MMap given the following cryptoprimitives:

- A probabilistic-iO scheme piO $= (\mathsf{piO.Obf}, \mathsf{piOEval})$.
- A fully-homomorphic encryption scheme

$$\text{FHE} = (\mathsf{FHE.Gen}, \mathsf{FHE.Enc}, \mathsf{FHE.Dec}, \mathsf{FHE.Eval}),$$

  such that the message space is $\mathbb{Z}_q$ for some prime $q = \text{poly}(\lambda)$ (λ being the security parameter).

- A dual mode NIZK argument system NIZK $= (\mathsf{NIZK.Setup}, \mathsf{NIZK.Prove}, \mathsf{NIZK.Verify})$ that is:
  - perfectly sound and extractable in the binding mode, and
  - perfectly witness indistinguishable and perfectly zero-knowledge in the hiding mode.
- 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 $y \in \mathcal{L}$.
- A pairing-free group $\mathbb{G}$ of prime order q.

In Appendix 6 we show that SXDH is hard over our proposed MMap construction if DDH is hard over the group $\mathbb{G}$. Subsequently, in Appendix 7, we show that for any $k \geq 2$, k-exponent-DDH (abbreviated as k-EDDH) is hard over our proposed MMap construction if k-EDDH is hard over the group $\mathbb{G}$. Note that for $k \geq 2$, $(k+1)$-exponent DDH implies a host of other assumptions such as k-Lin and $(k+1)$-power-DDH (PDDH).

5.1 Setup

The setup algorithm for our MMap construction takes as input the security parameter 1^λ and a second parameter 1^n for the degree of multilinearity. It samples two key-pairs for the FHE scheme as:

$$(\mathsf{pk}_0, \mathsf{sk}_0), (\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathsf{FHE.Gen}(1^\lambda).$$

It then uniformly samples $g_0 \leftarrow \mathbb{G}$ and $\gamma_1, \gamma_2, \gamma_3 \leftarrow \mathbb{Z}_q$ and sets:

$$g_1 = g_0^{\gamma_1} \quad , \quad g_2 = g_0^{\gamma_2} \quad , \quad g_3 = g_0^{\gamma_3} \quad .$$

It also samples a pair of binding NIZK CRS strings (along with the corresponding extraction trapdoors) as:

$$(\mathsf{crs}_0, \mathsf{t}_{\mathsf{ext},0}), (\mathsf{crs}_1, \mathsf{t}_{\mathsf{ext},1}) \leftarrow \mathsf{NIZK.Setup}(1^\lambda, \mathsf{binding}).$$

The languages for which statements are proven under these CRS strings are described subsequently in Section 5.2. Next the setup algorithm uniformly samples a total $(n+2)$ elements from the set $\mathcal{X}$ that are all non-members for the subset $\mathcal{L}$. More formally, it samples

$$y, z_{1,0}, z_{1,1}, z_{2,0}, z_{2,1}, \ldots, z_{n,0}, z_{n,1} \leftarrow \mathcal{X} \setminus \mathcal{L}.$$

Finally, the setup algorithm computes and outputs four probabilistically obfuscated circuits of the form

$$\bar{\mathsf{C}}_{\mathsf{Add}} = \mathsf{piO.Obf}(\mathsf{C}_{\mathsf{Add}}) \quad , \quad \bar{\mathsf{C}}_{\mathsf{Inv}} = \mathsf{piO.Obf}(\mathsf{C}_{\mathsf{Inv}}),$$
$$\bar{\mathsf{C}}_{\mathsf{Mult}} = \mathsf{piO.Obf}(\mathsf{C}_{\mathsf{Mult}}) \quad , \quad \bar{\mathsf{C}}_{\mathsf{ext}} = \mathsf{piO.Obf}(\mathsf{C}_{\mathsf{ext}})$$

where $\bar{\mathsf{C}}_{\mathsf{Add}}$, $\bar{\mathsf{C}}_{\mathsf{Inv}}$ $\bar{\mathsf{C}}_{\mathsf{Mult}}$, and $\bar{\mathsf{C}}_{\mathsf{ext}}$ are the circuits for adding, inverting, multiplying, and extracting from encodings at any given "non-zero" level. We describe these circuits in details subsequently. We only briefly mention here that these circuits embed the following elements that we want to keep secret for reasons relevant to the proof of security:

- The FHE secret keys sk_0 and sk_1.
- The NIZK extraction trapdoors $\mathsf{t}_{\mathsf{ext},0}$ and $\mathsf{t}_{\mathsf{ext},1}$.
- The tuple of exponents $(\gamma_1, \gamma_2, \gamma_3)$ and the tuple of group elements (g_0, g_1, g_2, g_3).

This is why these circuits are not made public as is. Instead, the setup algorithm only makes available piO-obfuscated versions of these circuit.

5.2 Encodings

We describe the procedure of encoding a plaintext element at any level $\mathbf{i}$ such that $\mathbf{0} \leq \mathbf{i} \leq \mathbf{n}$. In our construction, level-$\mathbf{0}$ encodings are treated slightly different from encodings at other "non-zero" levels, and are equipped with their own set of algorithms for encoding, manipulation, extraction and zero-testing. We informally mention these for the sake of completeness.

Level-Zero Encodings. We set the level-$\mathbf{0}$ encoding of a plaintext element $a \in \mathbb{Z}_q$ to be 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 5.3.

Extracting a level-$\mathbf{0}$ encoding $a \in \mathbb{Z}_q$ outputs g^a, where g is uniformly from the group $\mathbb{G}$ and is part of the public description of the MMap. As will be clear later, this is consistent with the extraction algorithm for any other level $\mathbf{i}$. Zero-testing follows trivially from the extraction algorithm.

Level-i Encodings. We now describe the procedure of encoding a plaintext element at any "non-zero" encoding 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 $\mathbf{i}$ is a tuple of the form $(\mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, \pi_0, \pi_1)$, where ct_0 and ct_1 are FHE encryptions of a tuple of the form:

$$(a_{0,0}, a_{1,0}, a_{2,0}, a_{3,0}, \mathbf{i}) \quad , \quad (a_{0,1}, a_{1,1}, a_{2,1}, a_{3,1}, \mathbf{i}).$$

under the public key-secret key pairs $(\mathsf{pk}_0, \mathsf{sk}_0)$ and $(\mathsf{pk}_1, \mathsf{sk}_1)$ respectively, and π_0 and π_1 are verifying proofs under crs_0 and crs_1,respectively, for statements described subsequently.

Normal and Oblique Representation. For $b \in \{0, 1\}$, the tuple $(a_{0,b}, a_{1,b}, a_{2,b}, a_{3,b}, \mathbf{i})$ is said to be in "normal form" if

$$(a_{0,b}, a_{1,b}, a_{2,b}, a_{3,b}, \mathbf{i}) = (a, 0, 0, 0, \mathbf{i})$$

for some $a \in \mathbb{Z}_q$. Otherwise, it is said to be in "oblique-form". Depending on the forms of the tuples underlying the FHE ciphertexts, we classify an encoding into one of three representations:

- **Normal representation:** Both tuples are in normal form.
- **Partially oblique representation:** Exactly one of the tuples is in normal form, while the other is in oblique form. Unless otherwise specified, we assume that the second tuple is in normal form.
- **Oblique representation:** Both tuples are in oblique form.

Consistency. Recall that the setup algorithm privately samples a tuple of group elements (g_0, g_1, g_2, g_3) and hardwires these into the MMap circuits that are described subsequently. We say that an encoding is "consistent" if the tuples underlying the FHE ciphertexts satisfy the following condition:

$$a_{0,0} + \sum_{\ell \in [3]} \gamma_\ell \cdot a_{\ell,0} = a_{0,1} + \sum_{\ell \in [3]} \gamma_\ell \cdot a_{\ell,1},$$

or equivalently, the following condition:

$$\prod_{\ell \in [0,3]} g_\ell^{a_{\ell,0}} = \prod_{\ell \in [0,3]} g_\ell^{a_{\ell,1}}.$$

Looking ahead, this will be used explicitly in the extraction algorithm of our construction.

NIZK Proof π_0. π_0 is a verifying NIZK proof of "normal representation". Informally, it proves under crs_0 that one of the following statements must be true:

1. Either $a_{0,0} = a_{0,1} = a$ and $a_{\ell,0} = a_{\ell,1} = 0$ for $\ell \in \{1, 2, 3\}$ and both FHE ciphertexts encrypt the same level $\mathbf{i}$ as in the encoding itself.
2. Or there exists some $j \in [n]$ such that $\mathbf{i}_j = 0$ and $z_{j,0} \in \mathcal{L}$.
3. Or there exists some $j \in [n]$ such that $\mathbf{i}_j = 1$ and $z_{j,1} \in \mathcal{L}$.

Formally, π_0 is a verifying NIZK proof under crs_0 of the **OR** relation R_0 defined below ($\mathcal{K}_{\text{FHE}}$ being the set of all valid key pairs under the FHE scheme):

Relation $\mathsf{R}_{\mathcal{L}}$:

$\mathsf{R}_{\mathcal{L}}(z, \mathsf{wit}_z) = 1$ if and only if $z \in \mathcal{L}$ with membership witness wit_z.

Relation $\mathsf{R}_{0,0}$:

$\mathsf{R}_{0,0}((\mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, \mathsf{pk}_0, \mathsf{pk}_1), \mathsf{wit}) = 1$ if and only if:

- **EITHER** $\mathsf{wit} = (\mathsf{sk}_0, \mathsf{sk}_1)$ *and* $(\mathsf{pk}_0, \mathsf{sk}_0), (\mathsf{pk}_1, \mathsf{sk}_1) \in \mathcal{K}_{\text{FHE}}$ *and*

$$\mathsf{FHE.Dec}(\mathsf{sk}_0, \mathsf{ct}_0) = \mathsf{FHE.Dec}(\mathsf{sk}_1, \mathsf{ct}_1) = (a, 0, 0, 0, \mathbf{i}) \text{ for some } a \in \mathbb{Z}_q.$$

- **OR** $\mathsf{wit} = (a, \mathsf{r}_0, \mathsf{r}_1)$ for some $a \in \mathbb{Z}_q$ *and* for each $b \in \{0, 1\}$, we have:

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

Relation $\mathsf{R}_{0,1}$:

$\mathsf{R}_{0,1}(\{z_{j,b}\}_{j\in[n], b\in\{0,1\}}, \mathbf{i}, \mathsf{wit}) = 1$ if and only if:

- $\mathsf{wit} = (j, \mathsf{wit}_z)$ for some $j \in [n]$ *and* $\mathsf{R}_{\mathcal{L}}(z_{j,\mathbf{i}_j}, \mathsf{wit}_z) = 1$.

Relation R_0:

$\mathsf{R}_0((\mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, \mathsf{pk}_0, \mathsf{pk}_1, \{z_{j,b}\}_{j\in[n], b\in\{0,1\}}), \mathsf{wit}) = 1$ if and only if:

- **EITHER** $\mathsf{R}_{0,0}((\mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, \mathsf{pk}_0, \mathsf{pk}_1), \mathsf{wit}) = 1$.
- **OR** $\mathsf{R}_{0,1}(\{z_{j,b}\}_{j\in[n], b\in\{0,1\}}, \mathbf{i}, \mathsf{wit}) = 1$.

NIZK Proof π_1. π_1 is a verifying NIZK proof of "consistency" that holds irrespective of whether the encoding is in normal representation or (partially) oblique representation. Informally, it proves under crs_1 that one of the following statements must be true with respect to the tuple of group elements (g_0, g_1, g_2, g_3) embedded inside the MMap circuits:

1. Either we have:
$$a_{0,0} + \sum_{\ell\in[3]} \gamma_\ell \cdot a_{\ell,0} = a_{0,1} + \sum_{\ell\in[3]} \gamma_\ell \cdot a_{\ell,1},$$
i.e., equivalently, we have:
$$\prod_{\ell\in[0,3]} g_\ell^{a_{\ell,0}} = \prod_{\ell\in[0,3]} g_\ell^{a_{\ell,1}}.$$

2. Or $y \in \mathcal{L}$.

Formally, π_1 is a verifying NIZK proof under crs_1 of the **OR** relation R_1 defined below ($\mathsf{R}_{\mathcal{L}}$ is as defined above):

Relation $\mathsf{R}_{1,0}$:

$\mathsf{R}_{1,0}((\mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, \mathsf{pk}_0, \mathsf{pk}_1), \mathsf{wit}) = 1$ if and only if:

- **EITHER** $\mathsf{wit} = (\mathsf{sk}_0, \mathsf{sk}_1)$ *and* $(\mathsf{pk}_0, \mathsf{sk}_0), (\mathsf{pk}_1, \mathsf{sk}_1) \in \mathcal{K}_{\text{FHE}}$ *and* for $b \in \{0, 1\}$, we have:

$$\mathsf{FHE.Dec}(\mathsf{sk}_b, \mathsf{ct}_b) = (a_{0,b}, a_{1,b}, a_{2,b}, a_{3,b}, \mathbf{i}),$$

  and we have:

$$\prod_{\ell \in [0,3]} g_\ell^{a_{\ell,0}} = \prod_{\ell \in [0,3]} g_\ell^{a_{\ell,1}}.$$

- **OR** $\mathsf{wit} = (\{a_{\ell,0}, a_{\ell,1}\}_{\ell \in [0,3]}, \mathsf{r}_0, \mathsf{r}_1)$ *and* for each $b \in \{0, 1\}$, we have

$$\mathsf{FHE.Enc}(\mathsf{pk}_b, (a_{0,b}, a_{1,b}, a_{2,b}, a_{3,b}, \mathbf{i}); \mathsf{r}_b) = \mathsf{ct}_b,$$

  and we have:

$$\prod_{\ell \in [0,3]} g_\ell^{a_{\ell,0}} = \prod_{\ell \in [0,3]} g_\ell^{a_{\ell,1}}.$$

Relation R_1:

$\mathsf{R}_1((\mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, \mathsf{pk}_0, \mathsf{pk}_1, y), \mathsf{wit}) = 1$ if and only if:

- **EITHER** $\mathsf{R}_{1,0}((\mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, \mathsf{pk}_0, \mathsf{pk}_1), \mathsf{wit}) = 1$.
- **OR** $\mathsf{wit} = \mathsf{wit}_y$ *and* $\mathsf{R}_{\mathcal{L}}(y, \mathsf{wit}_y) = 1$.

Validity. Finally, an encoding $(\mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, \pi_0, \pi_1)$ is said to be "valid" if both of the following hold simultaneously:

$$\mathsf{NIZK.Verify}(\mathsf{crs}_0, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, \{z_{j,b}\}_{j \in [n], b \in \{0,1\}}), \pi_0) = 1,$$

and

$$\mathsf{NIZK.Verify}(\mathsf{crs}_1, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, y), \pi_1) = 1.$$

Note that when $z_{j,b} \notin \mathcal{L}$ for each $(j, b) \in [n] \times \{0, 1\}$ and $y \notin \mathcal{L}$, any valid encoding must be in the normal representation, and hence must also be consistent. So having the additional proof π_1 might appear redundant. However, looking ahead, during certain hybrids in the proof of security, we will switch one or more of these elements from members to non-members, thereby allow encodings corresponding to either all levels or certain designated levels to be in oblique representation. In such a case, the proof π_1 would allow us to enforce consistency irrespective of whether the encoding is in the normal representation or in the oblique representation.

5.3 Addition and Inversion of Encodings

We now describe the procedure for adding two encodings, and for (additive) inversion of an encoding. Suppose we have two encodings at the same level $\mathbf{i}$ of the form:

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

Figure 2 details the operation of the encoding-addition circuit $\mathsf{C}_{\mathsf{Add}}$. Note that it embeds multiple secrets, including the FHE secret keys sk_0 and sk_1, as well as the extraction trapdoors $\mathsf{t}_{\mathsf{ext},0}$ and $\mathsf{t}_{\mathsf{ext},1}$ 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

$\underline{\mathsf{C}_{\mathsf{Add},\mathbf{FHE}}((\{a_{\ell,1}\}_{\ell\in[0,3]},\mathbf{i}_1),(\{a_{\ell,2}\}_{\ell\in[0,3]},\mathbf{i}_2))\textbf{:}}$

Output $\left(\{a_{\ell,1}+a_{\ell,2}\}_{\ell\in[0,3]},\mathbf{i}_1\right)$.

$\underline{\mathsf{C}_{\mathsf{Inv},\mathbf{FHE}}(\{a_\ell\}_{\ell\in[0,3]},\mathbf{i})\textbf{:}}$

Output $\left(\{-a_\ell \mod q\}_{\ell\in[0,3]},\mathbf{i}\right)$.

Figure 1: Circuits $\mathsf{C}_{\mathsf{Add},\mathrm{FHE}}$ and $\mathsf{C}_{\mathsf{Inv},\mathrm{FHE}}$

algorithm of the probabilistic iO scheme piO on it. The same holds for the encoding-inversion circuit C_{inv}, which is described in Figure 3

At a high level, we add the two input encodings by exploiting the fully-homomorphic nature of the encryption scheme. More concretely, we homomorphically evaluate the circuit $\mathsf{C}_{\mathsf{Add},\mathrm{FHE}}$ (described in Figure 1) on the corresponding ciphertext components of the two input encodings to generate the ciphertext components for the output encoding. We also generate proofs for normal representation and consistency of the output encoding using the tuple of secret keys $(\mathsf{sk}_0,\mathsf{sk}_1)$ as witness, unless otherwise dictated by the input encodings (in which case we use the extracted witnesses from the proofs in the input encodings to generate the proofs for the output encodings). The approach for inverting an input encoding is very similar, except that we homomorphically evaluate the circuit $\mathsf{C}_{\mathsf{Inv},\mathrm{FHE}}$ (also described in Figure 1) on the ciphertext components of the input encoding.

For technical reasons that are relevant to the proof of security, we check the following in both the addition and inversion circuits:

1. The validity of the proofs π_1 and π_2 for the language L that are provided as part of the input encodings.

2. Whether the encodings are in the normal representation as per the relation R_0 described earlier.

3. Whether the encodings are consistent as per the relation R_1 described earlier.

Note that validity is publicly verifiable, while verifying normal representation and consistency require knowledge of the secret keys sk_0 and sk_1.

The checks in steps 5 and 7 of $\mathsf{C}_{\mathsf{Add}}$ and $\mathsf{C}_{\mathsf{Inv}}$ are included for technical reasons that are relevant to the proof of security. In particular, we emphasize the following:

- Checking the "**If**" condition in step 7 of $\mathsf{C}_{\mathsf{Add}}$ (and $\mathsf{C}_{\mathsf{Inv}}$) requires the tuple of group elements (g_0,g_1,g_2,g_3) sampled at setup, which is hardwired into both circuits. Note, however, that the exponents α_1,α_2 and α_3 are not required, and are hence not hardwired into either circuit.

- When each element $z_{j,b}$ for $j\in[n]$ and $b\in\{0,1\}$ is a non-member for the language $\mathcal{L}$, under a binding crs_0, the "**If**" condition in step 5 of $\mathsf{C}_{\mathsf{Add}}$ (and $\mathsf{C}_{\mathsf{Inv}}$) is never satisfied. This follows from the perfect soundness guarantee of the NIZK proof system. However, the condition may be satisfied during some hybrid in the proof of security, when some element $z_{j,b}$ is "switched" to a member of $\mathcal{L}$.

- When the element y is a non-member for the language $\mathcal{L}$, under a binding crs_1, the "**If**" condition in step 7 of $\mathsf{C}_{\mathsf{Add}}$ (and $\mathsf{C}_{\mathsf{Inv}}$) is never satisfied. This again follows from the perfect soundness guarantee of the NIZK proof system. However, the condition may be satisfied during some hybrid in the proof of security, when the element y is "switched" to a member of $\mathcal{L}$.

```
$\underline{\mathsf{C}_{\mathsf{Add}}[\{\mathsf{sk}_b, \mathsf{pk}_b, \mathsf{crs}_b, \mathsf{t}_{\mathsf{ext},b}\}_{b\in\{0,1\}}, (g_0, g_1, g_2, g_3)]\ (\{(\mathsf{ct}_{0,k}, \mathsf{ct}_{1,k}, \mathbf{i}_k, \pi_{0,k}, \pi_{1,k})\}_{k\in\{1,2\}})}$:

1. Output $\perp$ if $\mathbf{i}_1 \neq \mathbf{i}_2$ or $\mathbf{i}_1 > \mathbf{n}$. Else, set $\mathbf{i} = \mathbf{i}_1$ and proceed.
2. Output $\perp$ if for any $k \in \{1,2\}$, we have
       $\mathsf{NIZK.Verify}(\mathsf{crs}_0, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_{0,k}, \mathsf{ct}_{1,k}, \mathbf{i}, \{z_{j,b}\}_{j\in[n],b\in\{0,1\}}), \pi_{0,k}) = 0.$
3. Output $\perp$ if for any $k \in \{1,2\}$, we have
       $\mathsf{NIZK.Verify}(\mathsf{crs}_1, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_{0,k}, \mathsf{ct}_{1,k}, \mathbf{i}, y), \pi_{1,k}) = 0.$
4. For $b \in \{0,1\}$, set: $\mathsf{ct}^*_b = \mathsf{FHE.Eval}(\mathsf{pk}_b, \mathsf{ct}_{b,1}, \mathsf{ct}_{b,2}, \mathsf{C}_{\mathsf{Add},\mathrm{FHE}})$.
5. If for some $k \in \{1,2\}$, $\mathsf{R}_{0,0}((\mathsf{ct}_{0,k}, \mathsf{ct}_{1,k}, \mathbf{i}, \mathsf{pk}_0, \mathsf{pk}_1), (\mathsf{sk}_0, \mathsf{sk}_1)) = 0$, then:
    (a) Extract $(j, \mathsf{wit}_z) = \mathsf{NIZK.Ext}(\mathsf{t}_{\mathsf{ext},0}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_{0,1}, \mathsf{ct}_{1,1}, \mathbf{i}, \{z_{j,b}\}_{j\in[n],b\in\{0,1\}}), \pi_{0,1})$.
    (b) If $\mathsf{R}_{\mathcal{L}}(z_{j,\mathbf{i}_j}, \mathsf{wit}_z) = 0$, output $\perp$.
    (c) Else, generate $\pi^*_0 \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}_0, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}^*_0, \mathsf{ct}^*_1, \mathbf{i}, \{z_{j,b}\}_{j\in[n],b\in\{0,1\}}), (j, \mathsf{wit}_z))$.
6. Else, generate $\pi^*_0 \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}^*_0, \mathsf{ct}^*_1, \mathbf{i}, \{z_{j,b}\}_{j\in[n],b\in\{0,1\}}), (\mathsf{sk}_0, \mathsf{sk}_1))$.
7. If for some $k \in \{1,2\}$, we have $\mathsf{R}_{1,0}((\mathsf{ct}_{0,k}, \mathsf{ct}_{1,k}, \mathbf{i}, \mathsf{pk}_0, \mathsf{pk}_1), (\mathsf{sk}_0, \mathsf{sk}_1)) = 0$, then:
    (a) Extract $\mathsf{wit}_y = \mathsf{NIZK.Ext}(\mathsf{t}_{\mathsf{ext},1}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_{0,1}, \mathsf{ct}_{1,1}, \mathbf{i}, y), \pi_{1,1})$.
    (b) If $\mathsf{R}_{\mathcal{L}}(y, \mathsf{wit}_y) = 0$, output $\perp$.
    (c) Else, generate $\pi^*_1 \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}_1, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}^*_0, \mathsf{ct}^*_1, \mathbf{i}, y), \mathsf{wit}_y)$.
8. Else, generate $\pi^*_1 \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}^*_0, \mathsf{ct}^*_1, \mathbf{i}, y), (\mathsf{sk}_0, \mathsf{sk}_1))$.
9. Output $(\mathsf{ct}^*_0, \mathsf{ct}^*_1, \mathbf{i}, \pi^*_0, \pi^*_1)$.
```

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

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:

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

At a high level, we multiply the two input encodings by again exploiting the fully-homomorphic nature of the encryption scheme. However, compared to addition and inversion, multiplication of two encodings is more involved and requires some careful decision-making regarding the format of the output encoding, as well as which witness to use when generating the output proof π^*_0. Based on the aforementioned observations, we divide the multiplication of encodings into two keys steps:

1. **Step-1:** Homomorphically evaluate the ciphertexts ct^*_0 and ct^*_1 corresponding to the output encoding.

2. **Step-2:** Generate the proof π^*_0 for the output encoding

We now present some detailed observations regarding the challenges for both steps:

```
$\underline{\mathsf{C}_{\mathsf{Inv}}[\{\mathsf{sk}_b, \mathsf{pk}_b, \mathsf{crs}_b, \mathsf{t}_{\mathsf{ext},b}\}_{b\in\{0,1\}}, (g_0, g_1, g_2, g_3)](\mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, \pi_0, \pi_1)}$:

    1. Output $\perp$ if $\mathbf{i} > \vec{1}_n$.
    2. Output $\perp$ if $\mathsf{NIZK.Verify}(\mathsf{crs}_0, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, \{z_{j,b}\}_{j\in[n],b\in\{0,1\}}), \pi_0) = 0$.
    3. Output $\perp$ if $\mathsf{NIZK.Verify}(\mathsf{crs}_1, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, y), \pi_1) = 0$.
    4. For $b \in \{0,1\}$, set: $\mathsf{ct}_b^* = \mathsf{FHE.Eval}(\mathsf{pk}_b, \mathsf{ct}_b, \mathsf{C}_{\mathsf{Inv,FHE}})$.
    5. If $\mathsf{R}_{0,0}((\mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, \mathsf{pk}_0, \mathsf{pk}_1), (\mathsf{sk}_0, \mathsf{sk}_1)) = 0$, then:
        (a) Extract $(j, \mathsf{wit}_z) = \mathsf{NIZK.Ext}(\mathsf{t}_{\mathsf{ext},0}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, \{z_{j,b}\}_{j\in[n],b\in\{0,1\}}), \pi_0)$.
        (b) If $\mathsf{R}_{\mathcal{L}}(z_{j,\mathbf{i}_j}, \mathsf{wit}_{z_j}) = 0$, output $\perp$.
        (c) Else, generate $\pi_0^* \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}_0, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0^*, \mathsf{ct}_1^*, \mathbf{i}, \{z_{j,b}\}_{j\in[n],b\in\{0,1\}}), (j, \mathsf{wit}_z))$.
    6. Else, generate $\pi_0^* \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0^*, \mathsf{ct}_1^*, \mathbf{i}, \{z_{j,b}\}_{j\in[n],b\in\{0,1\}}), (\mathsf{sk}_0, \mathsf{sk}_1))$.
    7. If $\mathsf{R}_{1,0}((\mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, \mathsf{pk}_0, \mathsf{pk}_1), (\mathsf{sk}_0, \mathsf{sk}_1)) = 0$, then:
        (a) Extract $\mathsf{wit}_y = \mathsf{NIZK.Ext}(\mathsf{t}_{\mathsf{ext},1}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, y), \pi_1)$.
        (b) If $\mathsf{R}_{\mathcal{L}}(y, \mathsf{wit}_y) = 0$, output $\perp$.
        (c) Else, generate $\pi_1^* \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}_1, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0^*, \mathsf{ct}_1^*, \mathbf{i}, y), \mathsf{wit}_y)$.
    8. Else, generate $\pi_1^* \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0^*, \mathsf{ct}_1^*, \mathbf{i}, y), (\mathsf{sk}_0, \mathsf{sk}_1))$.
    9. Output $(\mathsf{ct}_0^*, \mathsf{ct}_1^*, \mathbf{i}, \pi_0^*, \pi_1^*)$.
```

Figure 3: Circuit $\mathsf{C}_{\mathsf{Inv}}$

- Suppose that both input encodings are in the normal representation, i.e, they both contain verifying proofs of normal representation, using as witness the tuple of secret keys $(\mathsf{sk}_0, \mathsf{sk}_1)$. In this case, the output encoding upon multiplication is also in the normal representation, and can be computed without the knowledge of the secret exponents γ_1, γ_2 and γ_3. Additionally, the output proof π_0^* can be generated as a verifying proof of normal representation using the tuple of secret keys $(\mathsf{sk}_0, \mathsf{sk}_1)$ as witness.

- When exactly one of the input encodings is in oblique representation, then the output encoding upon multiplication can be computed in oblique representation without the knowledge of the secret exponents γ_1, γ_2 and γ_3. However, generating the proof π_0^* for the output encoding is not as straightforward. Suppose w.l.o.g. that the first input encoding in level $\mathbf{i}_1$ is in oblique representation, and consider the following scenarios:
  - Suppose that the first input encoding has a verifying proof of language membership of $z_{j,1}$ using witness (j, wit_z) such that $\mathbf{i}_{1,j} = 1$. In this case, we know that the output encoding must correspond to a level $\mathbf{i}^* = \mathbf{i}_1 + \mathbf{i}_2$ such that $\mathbf{i}_j^* = 1$. Hence, in this case, we can use the first circuit $\mathsf{C}_{\mathsf{Mult,FHE},0}$ for homomorphic evaluation of the output encoding in oblique representation, and we can re-use the witness (j, wit_z) extracted from the first input encoding to generate a verifying proof π_0^* for language membership of $z_{j,1}$ as part of the output encoding.
  - Now, suppose that the first input encoding has a verifying proof of language membership of $z_{j,0}$ using witness (j, wit_z) such that $\mathbf{i}_{1,j} = 0$. However, the output encoding may correspond to a level $\mathbf{i}^* = \mathbf{i}_1 + \mathbf{i}_2$ such that $\mathbf{i}_j^* = 1$. In such as case, we cannot re-use the witness (j, wit_z) extracted from the first input encoding to generate a verifying proof π_0^* for the output encoding, since we will need to prove language membership of $z_{j,1}$ and not $z_{j,0}$.

So, in the second case we need to use a different approach - We explicitly use the knowledge of the secret exponents γ_1, γ_2 and γ_3 to transform the output encoding back to normal representation. This allows us to generate the output proof π_0^* as a proof for normal representation using the tuple of secret keys $(\mathsf{sk}_0, \mathsf{sk}_1)$ as witness.

- Finally, consider the case when both input encodings are in the oblique representation. Here, there certain sub-cases to consider. Suppose w.l.o.g that the first ciphertext of the first encoding and the second ciphertext of the second encoding encrypt tuples in oblique representation, while the remaining ciphertexts encrypt tuples in normal representation. Again, in this sub-case, the output encoding upon multiplication can be computed in oblique representation without the knowledge of the secret exponents γ_1, γ_2 and γ_3. However, generating the proof π_0^* for the output encoding presents the same potential challenges as before. So we handle ciphertext generation in this sub-case (and other similar sub-cases) in the same way as the previous case.

  Alternatively, suppose w.l.o.g that the first ciphertext of the first encoding and the first ciphertext of the second encoding encrypt tuples in oblique representation, while the remaining ciphertexts encrypt tuples in normal representation. Now, the output encoding upon multiplication cannot be computed (in either normal or oblique representation) without the knowledge of the secret exponents γ_1, γ_2 and γ_3. Hence, in this sub-case (and other similar sub-cases), we explicitly use the knowledge of the secret exponents γ_1, γ_2 and γ_3 to transform the output encoding back to normal representation. The output proof π_0^* can now be generated as a proof for normal representation using the tuple of secret keys $(\mathsf{sk}_0, \mathsf{sk}_1)$ as witness.

$\underline{\mathsf{C}_{\mathsf{Mult},\mathbf{FHE}}((\{a_{\ell,1}\}_{\ell\in[0,3]}, \mathbf{i}_1), (\{a_{\ell,2}\}_{\ell\in[0,3]}, \mathbf{i}_2), (\{\gamma_\ell\}_{\ell\in[0,3]}, \vec{0}_n), (\mathsf{flag}_1, \mathsf{flag}_2, 0, 0, \vec{0}_n)))):}$

1. If $(a_{1,1}, a_{2,1}, a_{3,1}) = (0,0,0)$ and $\mathsf{flag}_2 = 1$, then output $((\{a_{0,1} \cdot a_{\ell,2}\}_{\ell\in[0,3]}, (\mathbf{i}_1 + \mathbf{i}_2)))$.
2. Else if $(a_{1,2}, a_{2,2}, a_{3,2}) = (0,0,0)$ and $\mathsf{flag}_1 = 1$, then output $((\{a_{0,2} \cdot a_{\ell,1}\}_{\ell\in[0,3]}, (\mathbf{i}_1 + \mathbf{i}_2)))$.
3. Else output $((a^*, 0, 0, 0, (\mathbf{i}_1 + \mathbf{i}_2)))$, where:
$$a^* = \prod_{\ell,\ell'\in[0,3]} a_{\ell,0} \cdot a_{\ell',0} \cdot \gamma_\ell \cdot \gamma_{\ell'}.$$

Figure 4: Circuits $\mathsf{C}_{\mathsf{Mult},\mathrm{FHE}}$

Based on the aforementioned observations, we design the circuit $\mathsf{C}_{\mathsf{Mult},\mathrm{FHE}}$ (described in Figure 4). This circuit is homomorphically evaluated on the corresponding FHE ciphertext components of the two input encodings to generate the FHE ciphertext component for the output product encoding. Note the following:

- The circuit $\mathsf{C}_{\mathsf{Mult},\mathrm{FHE}}$ takes as input two flag variables flag_1 and flag_2. For each $k \in \{1,2\}$, the flag variable flag_k is set as follows: if $\mathsf{flag}_k = 0$, then it indicates that the k^{th} encoding in level $\mathbf{i}_k$ is in (partially) oblique representation and has a verifying NIZK proof $\pi_{0,k}$ using (j_k, wit_z), where $\mathbf{i}_{k,j_k} = 0$ and wit_z is a language-membership witness for $zj_k, 0$. In all other cases, flag_k is set to 1.
- If at least one input tuple is in normal form and the other input has the flag variable set to 1, then the circuit $\mathsf{C}_{\mathsf{Mult},\mathrm{FHE}}$ does not require to use the secret exponents γ_1, γ_2 and γ_3. In this case, the output tuple may be either in normal form or in oblique form depending on the nature of the other input tuple. This is captured in Steps 1 and 2 of the circuit $\mathsf{C}_{\mathsf{Mult},\mathrm{FHE}}$.
- If both input tuples are in oblique representation, or if both input tuples have flag variables set to 0, or if exactly one input tuple is in oblique representation but has its flag set to 0, then the circuit $\mathsf{C}_{\mathsf{Mult},\mathrm{FHE}}$ forces the output to be in normal representation, irrespective of the representation of the input tuples. As shown in Step 3 of the circuit $\mathsf{C}_{\mathsf{Mult},\mathrm{FHE}}$, this requires explicitly using the secret exponents γ_1, γ_2 and γ_3.

For generating the output proof π_0^*, we use the following simple approach:

```
$\underline{\mathsf{C}_{\mathsf{Mult}}[\{\mathsf{sk}_b, \mathsf{pk}_b, \mathsf{crs}_b, \mathsf{t}_{\mathsf{ext},b}\}_{b\in\{0,1\}}, (g_0, \gamma_1, \gamma_2, \gamma_3)]\ (\{(\mathsf{ct}_{0,k}, \mathsf{ct}_{1,k}, \mathbf{i}_k, \pi_{0,k}, \pi_{1,k})\}_{k\in\{1,2\}})\textbf{:}}$

1. Output $\perp$ if $\mathbf{i}_1 + \mathbf{i}_2 > \vec{1}_n$.

2. Output $\perp$ if for any $k \in \{1,2\}$, we have
       $\mathsf{NIZK.Verify}(\mathsf{crs}_0, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_{0,k}, \mathsf{ct}_{1,k}, \mathbf{i}_k, \{z_{j,b}\}_{j\in[n],b\in\{0,1\}}), \pi_{0,k}) = 0.$

3. Output $\perp$ if for any $k \in \{1,2\}$, $\mathsf{NIZK.Verify}(\mathsf{crs}_1, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_{0,k}, \mathsf{ct}_{1,k}, \mathbf{i}_k, y), \pi_{1,k}) = 0$.

4. For $k \in \{1,2\}$ extract
       $\mathsf{wit}_k = \mathsf{NIZK.Ext}(\mathsf{t}_{\mathsf{ext},0}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_{0,k}, \mathsf{ct}_{1,k}, \mathbf{i}_k, \{z_{j,b}\}_{j\in[n],b\in\{0,1\}}), \pi_{0,k}),$
   and set
       $\mathsf{flag}_k = \begin{cases} 0 & \text{if } \mathsf{wit}_k = (j_k, \mathsf{wit}_{z,k}) \text{ and } \mathbf{i}_{k,j_k} = 0, \\ 1 & \text{otherwise.} \end{cases}$

5. For $b \in \{0,1\}$, set: $\mathsf{ct}^*_b = \mathsf{FHE.Eval}(\mathsf{pk}_b, \mathsf{ct}_{b,1}, \mathsf{ct}_{b,2}, \mathsf{ct}_{b,\gamma}, \mathsf{ct}_{b,\mathsf{flag}}, \mathsf{C}_{\mathsf{Mult,FHE}})$, where
       $\mathsf{ct}_{b,\gamma} = \mathsf{FHE.Enc}(\mathsf{pk}_b, (0, \gamma_1, \gamma_2, \gamma_3, \vec{0}_n)),$
       $\mathsf{ct}_{b,\mathsf{flag}} = \mathsf{FHE.Enc}(\mathsf{pk}_b, (\mathsf{flag}_1, \mathsf{flag}_2, 0, \vec{0}_n)).$

6. If there exists some $k \in 1, 2$ such that:
       $\mathsf{R}_{1,0}((\mathsf{ct}_{0,k}, \mathsf{ct}_{1,k}, \mathbf{i}_k, \mathsf{pk}_0, \mathsf{pk}_1), (\mathsf{sk}_0, \mathsf{sk}_1)) = 0$ and $\mathsf{wit}_k = (j_k, \mathsf{wit}_{z,k})$ such that $\mathbf{i}_{k,j_k} = 1,$
   then generate
       $\pi^*_0 \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}^*_0, \mathsf{ct}^*_1, (\mathbf{i}_1 + \mathbf{i}_2), \{z_{j,b}\}_{j\in[n],b\in\{0,1\}}), (j_k, \mathsf{wit}_{z,k})).$

7. Else, generate
       $\pi^*_0 \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}^*_0, \mathsf{ct}^*_1, (\mathbf{i}_1 + \mathbf{i}_2), \{z_{j,b}\}_{j\in[n],b\in\{0,1\}}), (\mathsf{sk}_0, \mathsf{sk}_1)).$

8. If for some $k \in \{1,2\}$, we have $\mathsf{R}_{1,0}((\mathsf{ct}_{0,k}, \mathsf{ct}_{1,k}, \mathbf{i}_k, \mathsf{pk}_0, \mathsf{pk}_1), (\mathsf{sk}_0, \mathsf{sk}_1)) = 0$, then:
   (a) Extract $\mathsf{wit}_y = \mathsf{NIZK.Ext}(\mathsf{t}_{\mathsf{ext},1}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_{0,1}, \mathsf{ct}_{1,1}, \mathbf{i}_k, y), \pi_{1,k})$.
   (b) If $\mathsf{R}_{\mathcal{L}}(y, \mathsf{wit}_y) = 0$, output $\perp$.
   (c) Else, generate $\pi^*_1 \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}_1, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}^*_0, \mathsf{ct}^*_1, (\mathbf{i}_1 + \mathbf{i}_2), y), \mathsf{wit}_y)$.

9. Else, generate $\pi^*_1 \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}^*_0, \mathsf{ct}^*_1, (\mathbf{i}_1 + \mathbf{i}_2), y), (\mathsf{sk}_0, \mathsf{sk}_1))$.

10. Output $(\mathsf{ct}^*_0, \mathsf{ct}^*_1, \mathbf{i}, \pi^*_0, \pi^*_1)$.
```

Figure 5: Circuit $\mathsf{C}_{\mathsf{Mult}}$

- Suppose that there exists an input encoding in oblique representation that has a verifying proof of language membership of $z_{j,1}$ using witness (j, wit_z) such that $\mathbf{i}_{1,j} = 1$. In this case, we know that the output encoding must correspond to a level $\mathbf{i}^* = \mathbf{i}_1 + \mathbf{i}_2$ such that $\mathbf{i}^*_j = 1$. Hence, in this case, we *extract and re-use* the witness (j, wit_z) from the first input encoding to generate a verifying proof π^*_0 for language membership of $z_{j,1}$ as part of the output encoding.

- In all other cases, homomorphically evaluating the circuit $\mathsf{C}_{\mathsf{Mult,FHE}}$ on the input encodings must, by design, output ciphertexts that are in normal representation. Hence, in all other cases, we use the tuple of secret keys $(\mathsf{sk}_0, \mathsf{sk}_1)$ to generate a verifying proof π^*_0 for normal representation.

Figure 5 details the operation of the encoding-inversion circuit $\mathsf{C}_{\mathsf{Mult}}$. Note that it again embeds multiple secrets, including the FHE secret keys sk_0 and sk_1, as well as the extraction trapdoors $\mathsf{t}_{\mathsf{ext},0}$ and $\mathsf{t}_{\mathsf{ext},1}$ 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.

Similar to the addition and inversion procedures described previously, the checks in steps 6 and 8 of $\mathsf{C}_{\mathsf{Mult}}$ are included for technical reasons that are relevant to the proof of security. In particular, we emphasize the following:

- When each element $z_{j,0}$ for $j \in [n]$ and $b \in \{0,1\}$ is a non-member for the language $\mathcal{L}$, then under a binding crs_0, the "**If**" condition in step 6 of $\mathsf{C}_{\mathsf{Mult}}$ is never satisfied. This follows from the perfect soundness guarantee of the NIZK proof system.

  Looking ahead, in certain hybrids of our proof of SXDH, we do allow the "**If**" condition in step 6 to be satisfiable. In these hybrids, for some $j \in [n]$, we deliberately switch either $z_{j,1}$ or both $z_{j,0}$ and $z_{j,1}$ in the public parameter from a non-member to a member for $\mathcal{L}$.

- Finally, when the element y is a non-member for the language $\mathcal{L}$, then under a binding crs_1, the "**If**" condition in step 8 of $\mathsf{C}_{\mathsf{Mult}}$ is never satisfied. This again follows from the perfect soundness guarantee of the NIZK proof system. Looking ahead, in certain hybrids of our proof of SXDH, the "**If**" condition may be satisfied when the element y is deliberately switched from a non-member to a member of $\mathcal{L}$.

5.5 Extraction and Zero-Testing

Extraction. We now describe the procedure for extracting a canonical string from an encoding. Suppose we have an encodings at the level $\mathbf{i}$ of the form:

$$(\mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, \pi_0, \pi_1).$$

The extraction circuit uses sk_0 and sk_1 to recover the plaintext elements $\{a_{\ell,0}, a_{\ell,1}\}_{\ell\in[0,3]}$ underlying the FHE ciphertexts ct_0 and ct_1, and provided that these are consistent as per relation R_1 described earlier, outputs

$$g^* = \prod_{\ell\in[0,3]} g_\ell^{a_{\ell,0}} = \prod_{\ell\in[0,3]} g_\ell^{a_{\ell,1}}.$$

Figure 6 details the operation of the extraction circuit $\mathsf{C}_{\mathsf{ext}}$. Note that it again embeds multiple secrets, including the FHE secret keys sk_0 and sk_1, as well as the extraction trapdoors $\mathsf{t}_{\mathsf{ext},0}$ and $\mathsf{t}_{\mathsf{ext},1}$ 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.

Once again, similar to the addition, inversion and multiplication procedures described previously, the checks in steps 6 and 7 of $\mathsf{C}_{\mathsf{ext}}$ are included for technical reasons that are relevant to the proof of security.

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 Proof of SXDH Hardness

In this section, we prove that solving SXDH is hard over our proposed MMap construction if solving DDH is hard over the group $\mathbb{G}$. More specifically we state and prove the following theorem:

Theorem 6.1. *The SXDH assumption holds over our proposed MMap construction provided that all of the following assumptions hold:*

- *The piO scheme is indistinguishability-secure against X-IND samplers.*
- *Membership in $\mathcal{L}$ is computationally hard to decide,*

```
$\underline{\mathsf{C}_{\mathsf{ext}}[\{\mathsf{sk}_b, \mathsf{pk}_b, \mathsf{crs}_b, \mathsf{t}_{\mathsf{ext},b}\}_{b\in\{0,1\}}, (g_0, g_1, g_2, g_3)](\mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, \pi_0, \pi_1)}$:

  1. Output $\perp$ if $\mathbf{i} > \vec{1}_n$.
  2. Output $\perp$ if $\mathsf{NIZK.Verify}(\mathsf{crs}_0, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, \{z_{j,b}\}_{j\in[n],b\in\{0,1\}}), \pi_0) = 0$.
  3. Output $\perp$ if $\mathsf{NIZK.Verify}(\mathsf{crs}_1, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, y), \pi_1) = 0$.
  4. For $b \in \{0,1\}$, recover $(\{a_{\ell,b}\}_{\ell\in[0,3]}, \mathbf{i}_b) = \mathsf{FHE.Dec}(\mathsf{sk}_b, \mathsf{ct}_b)$.
  5. Compute $g^* = \prod_{\ell\in[0,3]} g_\ell^{a_{\ell,0}}$.
  6. If $\mathsf{R}_{0,0}((\mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, \mathsf{pk}_0, \mathsf{pk}_1), (\mathsf{sk}_0, \mathsf{sk}_1)) = 0$, then:
      (a) Extract $(j, \mathsf{wit}_{z_j}) = \mathsf{NIZK.Ext}(\mathsf{t}_{\mathsf{ext},0}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, \{z_{j,b}\}_{j\in[n],b\in\{0,1\}}), \pi_0)$.
      (b) If $\mathsf{R}_{\mathcal{L}}(z_j, \mathsf{wit}_{z_j}) = 0$, output $\perp$.
  7. If $\mathsf{R}_{1,0}((\mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, \mathsf{pk}_0, \mathsf{pk}_1), (\mathsf{sk}_0, \mathsf{sk}_1)) = 0$, then:
      (a) Extract $\mathsf{wit}_y = \mathsf{NIZK.Ext}(\mathsf{t}_{\mathsf{ext},1}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, y), \pi_1)$.
      (b) If $\mathsf{R}_{\mathcal{L}}(y, \mathsf{wit}_y) = 0$, output $\perp$.
  8. Output $g^*$.
```

Figure 6: Circuit $\mathsf{C}_{\mathsf{ext}}$

- *Members of $\mathcal{L}$ have unique witnesses for membership.*
- *The dual-mode NIZK proof system satisfies CRS indistinguishability, perfect extractability in the binding mode, and perfect witness-indistinguishability in the hiding mode.*
- *The DDH assumption holds over the group $\mathbb{G}$.*

We begin by outlining the hybrids that are used in the proof. The hybrids are classified into two broad categories - outer hybrids and inner hybrids.

6.1 Outer Hybrids

We begin by describing the outer hybrids for our proof. Outer hybrid 0 corresponds to the game where the challenger provides the adversary with encodings of uniformly random elements in a level-set that is chosen by the adversary. The final outer hybrid corresponds to the game where the challenger provides the adversary with a valid SXDH instance, i.e., encodings of elements sampled according to the real SXDH distribution, in the same level-set chosen by the adversary. Before we describe the outer hybrids, we describe a few conditions that remain invariant throughout the outer hybrids:

- The NIZK CRS strings crs_0 and crs_1 are in binding mode, as in the real MMap scheme, in all the outer hybrids.
- The element y in the public parameter is a non-member for the language $\mathcal{L}$, as in the real MMap scheme, in all the outer hybrids.
- The challenge encodings provided to the adversary are *consistent* with respect to extraction (as formalized by the relation R_1 described earlier) in all the outer hybrids. However, as we shall see later, they may be switched from the normal to oblique representation and vice-versa.

Table 1 provides an overview of the outer hybrids, which we now describe in details.

Outer Hybrid	MMap Circuits	$z_{j,0}$	$z_{j,1}$	(g_0, g_1, g_2, g_3)	Challenge Encodings			
					SXDH/Random	Representation	Witness for π_0	Witness for π_1
0	C_{op}	$z_{j,0} \notin \mathcal{L}$	$z_{j,1} \notin \mathcal{L}$	Random	Random	Normal	$(\mathsf{sk}_0, \mathsf{sk}_1)$	$(\mathsf{sk}_0, \mathsf{sk}_1)$
1	$\widehat{\mathsf{C}}_{\mathsf{op}}$	$z_{j,0} \in \mathcal{L}$	$z_{j,1} \in \mathcal{L}$	Random	Random	Normal	$(j, \mathsf{wit}_{z_{j,1}})$	$(\mathsf{sk}_0, \mathsf{sk}_1)$
2	$\widehat{\mathsf{C}}_{\mathsf{op}}$	$z_{j,0} \in \mathcal{L}$	$z_{j,1} \in \mathcal{L}$	Random	Random	Partially oblique	$(j, \mathsf{wit}_{z_{j,1}})$	$(\mathsf{sk}_0, \mathsf{sk}_1)$
3	$\widehat{\mathsf{C}}_{\mathsf{op}}$	$z_{j,0} \in \mathcal{L}$	$z_{j,1} \in \mathcal{L}$	Random	Random	Oblique	$(j, \mathsf{wit}_{z_{j,1}})$	$(\mathsf{sk}_0, \mathsf{sk}_1)$
4	$\widehat{\widehat{\mathsf{C}}}_{\mathsf{op}}$	$z_{j,0} \notin \mathcal{L}$	$z_{j,1} \in \mathcal{L}$	Random	Random	Oblique	$(j, \mathsf{wit}_{z_{j,1}})$	$(\mathsf{sk}_0, \mathsf{sk}_1)$
5	$\widehat{\widehat{\mathsf{C}}}_{\mathsf{op}}$	$z_{j,0} \notin \mathcal{L}$	$z_{j,1} \in \mathcal{L}$	DDH	SXDH	Oblique	$(j, \mathsf{wit}_{z_{j,1}})$	$(\mathsf{sk}_0, \mathsf{sk}_1)$
6	$\widehat{\mathsf{C}}_{\mathsf{op}}$	$z_{j,0} \in \mathcal{L}$	$z_{j,1} \in \mathcal{L}$	DDH	SXDH	Oblique	$(j, \mathsf{wit}_{z_{j,1}})$	$(\mathsf{sk}_0, \mathsf{sk}_1)$
7	$\widehat{\mathsf{C}}_{\mathsf{op}}$	$z_{j,0} \in \mathcal{L}$	$z_{j,1} \in \mathcal{L}$	DDH	SXDH	Partially oblique	$(j, \mathsf{wit}_{z_{j,1}})$	$(\mathsf{sk}_0, \mathsf{sk}_1)$
8	$\widehat{\mathsf{C}}_{\mathsf{op}}$	$z_{j,0} \in \mathcal{L}$	$z_{j,1} \in \mathcal{L}$	DDH	SXDH	Normal	$(j, \mathsf{wit}_{z_{j,1}})$	$(\mathsf{sk}_0, \mathsf{sk}_1)$
9	C_{op}	$z_{j,0} \notin \mathcal{L}$	$z_{j,1} \notin \mathcal{L}$	DDH	SXDH	Normal	$(\mathsf{sk}_0, \mathsf{sk}_1)$	$(\mathsf{sk}_0, \mathsf{sk}_1)$
10	C_{op}	$z_{j,0} \notin \mathcal{L}$	$z_{j,1} \notin \mathcal{L}$	Random	SXDH	Normal	$(\mathsf{sk}_0, \mathsf{sk}_1)$	$(\mathsf{sk}_0, \mathsf{sk}_1)$

Table 1: Overview of the outer hybrids in the proof of SXDH. Changes between subsequent hybrids are highlighted in red. Throughout, crs_0 and crs_1 are binding, $y \notin \mathcal{L}$, and the challenge encodings are consistent with respect to extraction. We use the shorthands C_{op} and $\widehat{\mathsf{C}}_{\mathsf{op}}$ for the tuples $(\mathsf{C}_{\mathsf{Add}}, \mathsf{C}_{\mathsf{Inv}}, \mathsf{C}_{\mathsf{Mult}}, \mathsf{C}_{\mathsf{ext}})$ and $(\widehat{\mathsf{C}}_{\mathsf{Add}}, \widehat{\mathsf{C}}_{\mathsf{Inv}}, \widehat{\mathsf{C}}_{\mathsf{Mult}}, \widehat{\mathsf{C}}_{\mathsf{ext}})$, respectively, where the second set of circuits are described in detail subsequently.

Outer Hybrid 0. In this hybrid, the MMap is set up exactly as in the real scheme described earlier. Let (g_0, g_1, g_2, g_3) be the tuple of group elements hardwired into each of the MMap circuits, such that $g_\ell = g_0^{\gamma_\ell}$ for $\ell \in \{1, 2, 3\}$. Let μ be a uniformly sampled element in $\mathbb{Z}_q$. The SXDH adversary is provided with encodings of the tuple of elements:

$$(\alpha_0, \alpha_1, \alpha_2, \alpha_3) = (\mu, \mu \cdot \gamma_1, \mu \cdot \gamma_2, \mu \cdot \gamma_3),$$

where the encodings are generated in normal form (formalized by relation R_0 described earlier) corresponding to the level-set $\mathbf{i}$ chosen by the SXDH adversary. For ease of understanding, we explicitly lay out what the FHE ciphertexts in each of the encodings actually encrypt in Table 2. Along with the FHE encryptions, each encoding also contains a NIZK proof π_0 for normal representation under the binding crs_0, and a NIZK proof π_1 for consistency with respect to extraction under the binding crs_1.

Outer Hybrid 1. In this hybrid, we make the following alterations to the manner in which the MMap is generated:

1. Fix $j \in [n]$ such that for the challenge level-set $\mathbf{i}$, we have $\mathbf{i}_j = 1$. We switch the elements $z_{j,0}$ and $z_{j,1}$ in the public parameters from non-members to members for $\mathcal{L}$, i.e., we now sample $z_{j,0}, z_{j,1} \leftarrow \mathcal{L}$, along with their (unique) membership-witnesses $\mathsf{wit}_{z_{j,0}}$ and $\mathsf{wit}_{z_{j,1}}$, respectively.

Encoding	ct_0 encrypts	ct_1 encrypts	Witness for π_0	Witness for π_1
$[\alpha_0]$	$(\mu, 0, 0, 0, \mathbf{i})$	$(\mu, 0, 0, 0, \mathbf{i})$	$(\mathsf{sk}_0, \mathsf{sk}_1)$	$(\mathsf{sk}_0, \mathsf{sk}_1)$
$[\alpha_1]$	$(\mu \cdot \gamma_1, 0, 0, 0, \mathbf{i})$	$(\mu \cdot \gamma_1, 0, 0, 0, \mathbf{i})$	$(\mathsf{sk}_0, \mathsf{sk}_1)$	$(\mathsf{sk}_0, \mathsf{sk}_1)$
$[\alpha_2]$	$(\mu \cdot \gamma_2, 0, 0, 0, \mathbf{i})$	$(\mu \cdot \gamma_2, 0, 0, 0, \mathbf{i})$	$(\mathsf{sk}_0, \mathsf{sk}_1)$	$(\mathsf{sk}_0, \mathsf{sk}_1)$
$[\alpha_3]$	$(\mu \cdot \gamma_3, 0, 0, 0, \mathbf{i})$	$(\mu \cdot \gamma_3, 0, 0, 0, \mathbf{i})$	$(\mathsf{sk}_0, \mathsf{sk}_1)$	$(\mathsf{sk}_0, \mathsf{sk}_1)$

Table 2: Overview of Challenge Encodings in Outer Hybrid 0

2. We switch the circuits for addition, inversion, multiplication and extraction as follows:

$$\mathsf{C}_{\mathsf{Add}} \mapsto \widehat{\mathsf{C}}_{\mathsf{Add}} \quad , \quad \mathsf{C}_{\mathsf{Inv}} \mapsto \widehat{\mathsf{C}}_{\mathsf{Inv}},$$
$$\mathsf{C}_{\mathsf{Mult}} \mapsto \widehat{\mathsf{C}}_{\mathsf{Mult}} \quad , \quad \mathsf{C}_{\mathsf{ext}} \mapsto \widehat{\mathsf{C}}_{\mathsf{ext}},$$

The switched circuits are described in Figures 7, 8, 9, and 10, respectively. At a high level, we make the following alterations:

- We hardwire the witnesses $\mathsf{wit}_{z_{j,0}}$ and $\mathsf{wit}_{z_{j,1}}$ into the modified addition and inversion circuits $\widehat{\mathsf{C}}_{\mathsf{Add}}$ and $\widehat{\mathsf{C}}_{\mathsf{Inv}}$, and remove the extraction trapdoor $\mathsf{t}_{\mathsf{ext},0}$ from both these circuits. Instead of extracting the membership witnesses from the input encodings, we directly use these hardwired witnesses to generate proofs of membership for the output encodings.

- We hardwire the witness $\mathsf{wit}_{z_{j,1}}$ into the modified multiplication circuit $\widehat{\mathsf{C}}_{\mathsf{Mult}}$, and remove the extraction trapdoor $\mathsf{t}_{\mathsf{ext},0}$ from it. Instead of extracting the membership witness from the input encoding, we directly use the hardwired witness to generate proofs of membership for the output encodings. Note that we only need one witness for the multiplication circuit; the witness $\mathsf{wit}_{z_{j,0}}$ is not used (the reader may observe that the original multiplication circuit would not have extracted $\mathsf{wit}_{z_{j,0}}$ either).

- Finally, we remove the extraction trapdoor $\mathsf{t}_{\mathsf{ext},0}$ from the modified extraction circuit $\widehat{\mathsf{C}}_{\mathsf{ext}}$

The following items are worth noting:

1. By switching both $z_{j,0}$ and $z_{j,1}$ to members of $\mathcal{L}$, we effectively allow valid encodings in *any* level to be represented using the oblique representation. In particular, valid oblique encodings in any level set $\mathbf{i}'$ can, in theory, prove membership of $z_{j,\mathbf{i}'_j}$ using the corresponding witness $\mathsf{wit}_{z_{\mathbf{i}'_j}}$.

2. The element y continues to be a non-member for $\mathcal{L}$ and crs_1 is still generated in the binding mode. Hence, any valid encoding must still satisfy consistency as per relation $\mathsf{R}_{1,0}$ described earlier.

Additionally, we make the following change to the manner in which the challenge encodings are generated: for each challenge encoding, the NIZK proof π_0 now proves (under the binding crs_0) that $z_{j,1} \in \mathcal{L}$ as opposed to proving that the encoding is in the normal representation. This is explicitly described in Table 3.

$\widehat{\mathsf{C}}_{\mathsf{Add}}[\{\mathsf{sk}_b, \mathsf{pk}_b, \mathsf{crs}_b\}_{b\in\{0,1\}}, \{\mathsf{wit}_{z_{j,b}}\}_{b\in\{0,1\}}, \mathsf{t}_{\mathsf{ext},1}, (g_0, g_1, g_2, g_3)]\left(\{(\mathsf{ct}_{0,k}, \mathsf{ct}_{1,k}, \mathbf{i}_k, \pi_{0,k}, \pi_{1,k})\}_{k\in\{1,2\}}\right)$:

1. Output $\perp$ if $\mathbf{i}_1 \neq \mathbf{i}_2$ or $\mathbf{i}_1 > \mathbf{n}$. Else, set $\mathbf{i} = \mathbf{i}_1$ and proceed.
2. Output $\perp$ if for any $k \in \{1, 2\}$, we have
$$\mathsf{NIZK.Verify}(\mathsf{crs}_0, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_{0,k}, \mathsf{ct}_{1,k}, \mathbf{i}, \{z_{j,b}\}_{j\in[n],b\in\{0,1\}}), \pi_{0,k}) = 0.$$
3. Output $\perp$ if for any $k \in \{1, 2\}$, we have
$$\mathsf{NIZK.Verify}(\mathsf{crs}_1, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_{0,k}, \mathsf{ct}_{1,k}, \mathbf{i}, y), \pi_{1,k}) = 0.$$
4. For $b \in \{0, 1\}$, set: $\mathsf{ct}^*_b = \mathsf{FHE.Eval}(\mathsf{pk}_b, \mathsf{ct}_{b,1}, \mathsf{ct}_{b,2}, \mathsf{C}_{\mathsf{Add,FHE}})$.
5. **If** for some $k \in \{1, 2\}$, we have $\mathsf{R}_{0,0}((\mathsf{ct}_{0,k}, \mathsf{ct}_{1,k}, \mathbf{i}, \mathsf{pk}_0, \mathsf{pk}_1), (\mathsf{sk}_0, \mathsf{sk}_1)) = 0$, then:
   (a) // Omitted, depends on $\mathsf{t}_{\mathsf{ext},0}$.
   (b) Generate $\pi^*_0 \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}_0, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}^*_0, \mathsf{ct}^*_1, \mathbf{i}, \{z_{j,b}\}_{j\in[n],b\in\{0,1\}}), (j, \mathsf{wit}_{z_{j,\mathbf{i}_j}}))$.
6. **Else**, generate $\pi^*_0 \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}^*_0, \mathsf{ct}^*_1, \mathbf{i}, \{z_{j,b}\}_{j\in[n],b\in\{0,1\}}), (\mathsf{sk}_0, \mathsf{sk}_1))$.
7. **If** for some $k \in \{1, 2\}$, we have $\mathsf{R}_{1,0}((\mathsf{ct}_{0,k}, \mathsf{ct}_{1,k}, \mathbf{i}, \mathsf{pk}_0, \mathsf{pk}_1), (\mathsf{sk}_0, \mathsf{sk}_1)) = 0$, then:
   (a) Extract $\mathsf{wit}_y = \mathsf{NIZK.Ext}(\mathsf{t}_{\mathsf{ext},1}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_{0,1}, \mathsf{ct}_{1,1}, \mathbf{i}, y), \pi_{1,1})$.
   (b) **If** $\mathsf{R}_{\mathcal{L}}(y, \mathsf{wit}_y) = 0$, output $\perp$.
   (c) **Else**, generate $\pi^*_1 \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}_1, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}^*_0, \mathsf{ct}^*_1, \mathbf{i}, y), \mathsf{wit}_y)$.
8. **Else**, generate $\pi^*_1 \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}^*_0, \mathsf{ct}^*_1, \mathbf{i}, y), (\mathsf{sk}_0, \mathsf{sk}_1))$.
9. Output $(\mathsf{ct}^*_0, \mathsf{ct}^*_1, \mathbf{i}, \pi^*_0, \pi^*_1)$.

Figure 7: Circuit $\widehat{\mathsf{C}}_{\mathsf{Add}}$

Encoding	ct_0 **encrypts**	ct_1 **encrypts**	**Witness for** π_0	**Witness for** π_1
$[\alpha_0]$	$(\mu, 0, 0, 0, \mathbf{i})$	$(\mu, 0, 0, 0, \mathbf{i})$	$(j, \mathsf{wit}_{z_{j,1}})$	$(\mathsf{sk}_0, \mathsf{sk}_1)$
$[\alpha_1]$	$(\mu \cdot \gamma_1, 0, 0, 0, \mathbf{i})$	$(\mu \cdot \gamma_1, 0, 0, 0, \mathbf{i})$	$(j, \mathsf{wit}_{z_{j,1}})$	$(\mathsf{sk}_0, \mathsf{sk}_1)$
$[\alpha_2]$	$(\mu \cdot \gamma_2, 0, 0, 0, \mathbf{i})$	$(\mu \cdot \gamma_2, 0, 0, 0, \mathbf{i})$	$(j, \mathsf{wit}_{z_{j,1}})$	$(\mathsf{sk}_0, \mathsf{sk}_1)$
$[\alpha_3]$	$(\mu \cdot \gamma_3, 0, 0, 0, \mathbf{i})$	$(\mu \cdot \gamma_3, 0, 0, 0, \mathbf{i})$	$(j, \mathsf{wit}_{z_{j,1}})$	$(\mathsf{sk}_0, \mathsf{sk}_1)$

Table 3: Overview of Challenge Encodings in Outer Hybrid 1

Outer Hybrid 2. This hybrid is identical to the outer hybrid 1, except that the challenge SXDH encodings are no longer in normal form. In particular, the first FHE ciphertext in each encoding now encrypts an oblique representation of the underlying plaintext element. For ease of understanding, we explicitly lay out what the FHE ciphertexts in each of the encodings actually encrypt in Table 4. As in outer hybrid 1, each encoding also contains a NIZK proof π_0 for $z_{j,1} \in \mathcal{L}$ under the binding crs_0, and a NIZK proof π_1 for consistency under the binding crs_1. The changes from outer

```
$\widehat{\mathsf{C}}_{\mathsf{Inv}}[\{\mathsf{sk}_b, \mathsf{pk}_b, \mathsf{crs}_b\}_{b\in\{0,1\}}, \{\mathsf{wit}_{z_{j,b}}\}_{b\in\{0,1\}}, \mathsf{t}_{\mathsf{ext},1}, (g_0, g_1, g_2, g_3)](\mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, \pi_0, \pi_1)$:

1. Output $\perp$ if $\mathbf{i} > \vec{1}_n$.
2. Output $\perp$ if $\mathsf{NIZK.Verify}(\mathsf{crs}_0, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, \{z_{j,b}\}_{j\in[n],b\in\{0,1\}}), \pi_0) = 0$.
3. Output $\perp$ if $\mathsf{NIZK.Verify}(\mathsf{crs}_1, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, y), \pi_1) = 0$.
4. For $b \in \{0,1\}$, set: $\mathsf{ct}_b^* = \mathsf{FHE.Eval}(\mathsf{pk}_b, \mathsf{ct}_b, \mathsf{C}_{\mathsf{Inv},\mathrm{FHE}})$.
5. If $\mathsf{R}_{0,0}((\mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, \mathsf{pk}_0, \mathsf{pk}_1), (\mathsf{sk}_0, \mathsf{sk}_1)) = 0$, then:
    (a) // Omitted, depends on $\mathsf{t}_{\mathsf{ext},0}$.
    (b) Generate $\pi_0^* \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}_0, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0^*, \mathsf{ct}_1^*, \mathbf{i}, \{z_{j,b}\}_{j\in[n],b\in\{0,1\}}), (j, \mathsf{wit}_{z_{j,\mathbf{i}_j}}))$.
6. Else, generate $\pi_0^* \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0^*, \mathsf{ct}_1^*, \mathbf{i}, \{z_{j,b}\}_{j\in[n],b\in\{0,1\}}), (\mathsf{sk}_0, \mathsf{sk}_1))$.
7. If $\mathsf{R}_{1,0}((\mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, \mathsf{pk}_0, \mathsf{pk}_1), (\mathsf{sk}_0, \mathsf{sk}_1)) = 0$, then:
    (a) Extract $\mathsf{wit}_y = \mathsf{NIZK.Ext}(\mathsf{t}_{\mathsf{ext},1}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, y), \pi_1)$.
    (b) If $\mathsf{R}_{\mathcal{L}}(y, \mathsf{wit}_y) = 0$, output $\perp$.
    (c) Else, generate $\pi_1^* \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}_1, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0^*, \mathsf{ct}_1^*, \mathbf{i}, y), \mathsf{wit}_y)$.
8. Else, generate $\pi_1^* \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0^*, \mathsf{ct}_1^*, \mathbf{i}, y), (\mathsf{sk}_0, \mathsf{sk}_1))$.
9. Output $(\mathsf{ct}_0^*, \mathsf{ct}_1^*, \mathbf{i}, \pi_0^*, \pi_1^*)$.
```

Figure 8: Circuit $\widehat{\mathsf{C}}_{\mathsf{Inv}}$

Encoding	ct_0 **encrypts**	ct_1 **encrypts**	**Witness for** π_0	**Witness for** π_1
$[\alpha_0]$	$(\mu, 0, 0, 0, \mathbf{i})$	$(\mu, 0, 0, 0, \mathbf{i})$	$(j, \mathsf{wit}_{z_{j,1}})$	$(\mathsf{sk}_0, \mathsf{sk}_1)$
$[\alpha_1]$	$(0, \mu, 0, 0, \mathbf{i})$	$(\mu \cdot \gamma_1, 0, 0, 0, \mathbf{i})$	$(j, \mathsf{wit}_{z_{j,1}})$	$(\mathsf{sk}_0, \mathsf{sk}_1)$
$[\alpha_2]$	$(0, 0, \mu, 0, \mathbf{i})$	$(\mu \cdot \gamma_2, 0, 0, 0, \mathbf{i})$	$(j, \mathsf{wit}_{z_{j,1}})$	$(\mathsf{sk}_0, \mathsf{sk}_1)$
$[\alpha_3]$	$(0, 0, 0, \mu, \mathbf{i})$	$(\mu \cdot \gamma_3, 0, 0, 0, \mathbf{i})$	$(j, \mathsf{wit}_{z_{j,1}})$	$(\mathsf{sk}_0, \mathsf{sk}_1)$

Table 4: Overview of Challenge Encodings in Outer Hybrid 2

hybrid 1 are highlighted in red.

Outer Hybrid 3. This hybrid is identical to the outer hybrid 2, except that the challenge SXDH encodings are now entirely in oblique form. In particular, the second FHE ciphertext in each encoding now also encrypts an oblique representation of the underlying plaintext element. Each encoding continues to have a NIZK proof π_0 for $z_{j,1} \in \mathcal{L}$ under the binding crs_0, and a NIZK proof π_1 for consistency with respect to extraction under the binding crs_1. For ease of understanding, we explicitly lay out what the FHE ciphertexts in each of the encodings actually encrypt in Table 5. The changes from outer hybrid 2 are highlighted in red.

$\widehat{\mathsf{C}}_{\mathsf{Mult}}[\{\mathsf{sk}_b, \mathsf{pk}_b, \mathsf{crs}_b\}_{b\in\{0,1\}}, \mathsf{wit}_{z_{j,1}}, \mathsf{t}_{\mathsf{ext},1}, (g_0, \gamma_1, \gamma_2, \gamma_3)]\left(\{(\mathsf{ct}_{0,k}, \mathsf{ct}_{1,k}, \mathbf{i}_k, \pi_{0,k}, \pi_{1,k})\}_{k\in\{1,2\}}\right)$:

1. Output $\perp$ if $\mathbf{i}_1 + \mathbf{i}_2 > \vec{1}_n$.

2. Output $\perp$ if for any $k \in \{1,2\}$, we have
$$\mathsf{NIZK.Verify}(\mathsf{crs}_0, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_{0,k}, \mathsf{ct}_{1,k}, \mathbf{i}_k, \{z_{j,b}\}_{j\in[n],b\in\{0,1\}}), \pi_{0,k}) = 0.$$

3. Output $\perp$ if for any $k \in \{1,2\}$, $\mathsf{NIZK.Verify}(\mathsf{crs}_1, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_{0,k}, \mathsf{ct}_{1,k}, \mathbf{i}_k, y), \pi_{1,k}) = 0$.

4. For $k \in \{1,2\}$ set
$$\mathsf{flag}_k = \begin{cases} 0 & \text{if } \mathbf{i}_{k,j} = 0, \\ 1 & \text{otherwise.} \end{cases}$$

   // Omitted use of $\mathsf{t}_{\mathsf{ext},0}$ above.

5. For $b \in \{0,1\}$, set: $\mathsf{ct}_b^* = \mathsf{FHE.Eval}(\mathsf{pk}_b, \mathsf{ct}_{b,1}, \mathsf{ct}_{b,2}, \mathsf{ct}_{b,\gamma}, \mathsf{ct}_{b,\mathsf{flag}}, \mathsf{C}_{\mathsf{Mult},\mathrm{FHE}})$, where
$$\mathsf{ct}_{b,\gamma} = \mathsf{FHE.Enc}(\mathsf{pk}_b, (0, \gamma_1, \gamma_2, \gamma_3, \vec{0}_n)),$$
$$\mathsf{ct}_{b,\mathsf{flag}} = \mathsf{FHE.Enc}(\mathsf{pk}_b, (\mathsf{flag}_1, \mathsf{flag}_2, 0, \vec{0}_n)).$$

6. **If** there exists some $k \in 1,2$ such that:
$$\mathsf{R}_{1,0}((\mathsf{ct}_{0,k}, \mathsf{ct}_{1,k}, \mathbf{i}_k, \mathsf{pk}_0, \mathsf{pk}_1), (\mathsf{sk}_0, \mathsf{sk}_1)) = 0 \text{ and } \mathbf{i}_{k,j} = 1,$$
   then generate
$$\pi_0^* \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0^*, \mathsf{ct}_1^*, (\mathbf{i}_1 + \mathbf{i}_2), \{z_{j,b}\}_{j\in[n],b\in\{0,1\}}), (j_k, \mathsf{wit}_{z,k})).$$

7. **Else**, generate
$$\pi_0^* \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0^*, \mathsf{ct}_1^*, (\mathbf{i}_1 + \mathbf{i}_2), \{z_{j,b}\}_{j\in[n],b\in\{0,1\}}), (\mathsf{sk}_0, \mathsf{sk}_1)).$$

8. **If** for some $k \in \{1,2\}$, we have $\mathsf{R}_{1,0}((\mathsf{ct}_{0,k}, \mathsf{ct}_{1,k}, \mathbf{i}_k, \mathsf{pk}_0, \mathsf{pk}_1), (\mathsf{sk}_0, \mathsf{sk}_1)) = 0$, then:
   (a) Extract $\mathsf{wit}_y = \mathsf{NIZK.Ext}(\mathsf{t}_{\mathsf{ext},1}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_{0,1}, \mathsf{ct}_{1,1}, \mathbf{i}_k, y), \pi_{1,k})$.
   (b) **If** $\mathsf{R}_{\mathcal{L}}(y, \mathsf{wit}_y) = 0$, output $\perp$.
   (c) **Else**, generate $\pi_1^* \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}_1, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0^*, \mathsf{ct}_1^*, (\mathbf{i}_1 + \mathbf{i}_2), y), \mathsf{wit}_y)$.

9. **Else**, generate $\pi_1^* \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0^*, \mathsf{ct}_1^*, (\mathbf{i}_1 + \mathbf{i}_2), y), (\mathsf{sk}_0, \mathsf{sk}_1))$.

10. Output $(\mathsf{ct}_0^*, \mathsf{ct}_1^*, \mathbf{i}, \pi_0^*, \pi_1^*)$.

Figure 9: Circuit $\widehat{\mathsf{C}}_{\mathsf{Mult}}$

Outer Hybrid 4. In this hybrid, we make the following alterations to the manner in which the MMap is generated:

1. We switch back the element $z_{j,0}$ in the public parameters from a member to a non-member for $\mathcal{L}$, i.e., we now sample $z_{j,0} \leftarrow \mathcal{X} \setminus \mathcal{L}$. The element $z_{j,1}$ continues to be a member for $\mathcal{L}$, i.e., we continue to sample $z_{j,1} \leftarrow \mathcal{L}$, along with the (unique) membership-witness $\mathsf{wit}_{z_{j,1}}$.

2. We switch the circuits for addition, inversion and multiplication as follows:
$$\widehat{\mathsf{C}}_{\mathsf{Add}} \mapsto \widehat{\widehat{\mathsf{C}}}_{\mathsf{Add}} \quad , \quad \widehat{\mathsf{C}}_{\mathsf{Inv}} \mapsto \widehat{\widehat{\mathsf{C}}}_{\mathsf{Inv}} \quad , \quad \widehat{\mathsf{C}}_{\mathsf{Mult}} \mapsto \widehat{\widehat{\mathsf{C}}}_{\mathsf{Mult}}.$$

```
$\widehat{\mathsf{C}}_{\mathsf{ext}}[\{\mathsf{sk}_b, \mathsf{pk}_b, \mathsf{crs}_b\}_{b\in\{0,1\}}, \mathsf{t}_{\mathsf{ext},1}, (g_0, g_1, g_2, g_3)](\mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, \pi_0, \pi_1)$:

1. Output $\perp$ if $\mathbf{i} > \vec{1}_n$.
2. Output $\perp$ if $\mathsf{NIZK.Verify}(\mathsf{crs}_0, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, \{z_{j,b}\}_{j\in[n],b\in\{0,1\}}), \pi_0) = 0$.
3. Output $\perp$ if $\mathsf{NIZK.Verify}(\mathsf{crs}_1, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, y), \pi_1) = 0$.
4. For $b \in \{0,1\}$, recover $(\{a_{\ell,b}\}_{\ell\in[0,3]}, \mathbf{i}_b) = \mathsf{FHE.Dec}(\mathsf{sk}_b, \mathsf{ct}_b)$.
5. Compute $g^* = \prod_{\ell\in[0,3]} g_\ell^{a_{\ell,0}}$.
6. // Omitted, depends on $\mathsf{t}_{\mathsf{ext},0}$.
7. If $\mathsf{R}_{1,0}((\mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, \mathsf{pk}_0, \mathsf{pk}_1), (\mathsf{sk}_0, \mathsf{sk}_1)) = 0$, then:
    (a) Extract $\mathsf{wit}_y = \mathsf{NIZK.Ext}(\mathsf{t}_{\mathsf{ext},1}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, y), \pi_1)$.
    (b) If $\mathsf{R}_{\mathcal{L}}(y, \mathsf{wit}_y) = 0$, output $\perp$.
8. Output $g^*$.
```

Figure 10: Circuit $\widehat{\mathsf{C}}_{\mathsf{ext}}$

Encoding	ct_0 **encrypts**	ct_1 **encrypts**	**Witness for** π_0	**Witness for** π_1
$[\alpha_0]$	$(\mu, 0, 0, 0, \mathbf{i})$	$(\mu, 0, 0, 0, \mathbf{i})$	$(j, \mathsf{wit}_{z_{j,1}})$	$(\mathsf{sk}_0, \mathsf{sk}_1)$
$[\alpha_1]$	$(0, \mu, 0, 0, \mathbf{i})$	$(0, \mu, 0, 0, \mathbf{i})$	$(j, \mathsf{wit}_{z_{j,1}})$	$(\mathsf{sk}_0, \mathsf{sk}_1)$
$[\alpha_2]$	$(0, 0, \mu, 0, \mathbf{i})$	$(0, 0, \mu, 0, \mathbf{i})$	$(j, \mathsf{wit}_{z_{j,1}})$	$(\mathsf{sk}_0, \mathsf{sk}_1)$
$[\alpha_3]$	$(0, 0, 0, \mu, \mathbf{i})$	$(0, 0, 0, \mu, \mathbf{i})$	$(j, \mathsf{wit}_{z_{j,1}})$	$(\mathsf{sk}_0, \mathsf{sk}_1)$

Table 5: Overview of Challenge Encodings in Outer Hybrid 3

The switched circuits for addition, inversion and multiplication are described in Figures 11, 12, and 14, respectively. The following are worth observing:

1. The modified circuits (in particular, the addition and inversion circuits $\widehat{\widehat{\mathsf{C}}}_{\mathsf{Add}}$ and $\widehat{\widehat{\mathsf{C}}}_{\mathsf{Inv}}$) only have the witness $\mathsf{wit}_{z_{j,1}}$ for the membership of $z_{j,1}$ in $\mathcal{L}$ hardwired into them. Since $z_{j,0}$ is no longer a member, it does not have a membership witness.

2. The modified multiplication circuit $\widehat{\widehat{\mathsf{C}}}_{\mathsf{Mult}}$ no longer has the exponents γ_1, γ_2 and γ_3 hardwired into it. It only has the group elements g_0, g_1, g_2 and g_3 hardwired, similar to the addition and inversion circuits.

3. The modified multiplication circuit $\widehat{\widehat{\mathsf{C}}}_{\mathsf{Mult}}$ in turn homomorphically evaluates a modified circuit $\widehat{\widehat{\mathsf{C}}}_{\mathsf{Mult,FHE}}$, described in Figure 13.

Finally, note that the element y continues to be a non-member for $\mathcal{L}$ and crs_1 is still generated in the binding mode. Hence, any valid encoding must still satisfy consistency with respect to extraction.

```
$\hat{\hat{\mathsf{C}}}_{\mathsf{Add}}[\{\mathsf{sk}_b, \mathsf{pk}_b, \mathsf{crs}_b\}_{b\in\{0,1\}}, \mathsf{wit}_{z_{j,1}}, \mathsf{t}_{\mathsf{ext},1}, (g_0, g_1, g_2, g_3)] \left(\{(\mathsf{ct}_{0,k}, \mathsf{ct}_{1,k}, \mathbf{i}_k, \pi_{0,k}, \pi_{1,k})\}_{k\in\{1,2\}}\right)$:

1. Output $\perp$ if $\mathbf{i}_1 \neq \mathbf{i}_2$ or $\mathbf{i}_1 > \mathbf{n}$. Else, set $\mathbf{i} = \mathbf{i}_1$ and proceed.

2. Output $\perp$ if for any $k \in \{1,2\}$, we have
   $$\mathsf{NIZK.Verify}(\mathsf{crs}_0, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_{0,k}, \mathsf{ct}_{1,k}, \mathbf{i}, \{z_{j,b}\}_{j\in[n],b\in\{0,1\}}), \pi_{0,k}) = 0.$$

3. Output $\perp$ if for any $k \in \{1,2\}$, we have
   $$\mathsf{NIZK.Verify}(\mathsf{crs}_1, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_{0,k}, \mathsf{ct}_{1,k}, \mathbf{i}, y), \pi_{1,k}) = 0.$$

4. For $b \in \{0,1\}$, set: $\mathsf{ct}^*_b = \mathsf{FHE.Eval}(\mathsf{pk}_b, \mathsf{ct}_{b,1}, \mathsf{ct}_{b,2}, \mathsf{C}_{\mathsf{Add},\mathrm{FHE}})$.

5. If for some $k \in \{1,2\}$, we have $\mathsf{R}_{0,0}((\mathsf{ct}_{0,k}, \mathsf{ct}_{1,k}, \mathbf{i}, \mathsf{pk}_0, \mathsf{pk}_1), (\mathsf{sk}_0, \mathsf{sk}_1)) = 0$, then:
   (a) // Omitted, depends on $\mathsf{t}_{\mathsf{ext},0}$.
   (b) Generate $\pi^*_0 \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}_0, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}^*_0, \mathsf{ct}^*_1, \mathbf{i}, \{z_{j,b}\}_{j\in[n],b\in\{0,1\}}), (j, \mathsf{wit}_{z_{j,1}}))$.

6. Else, generate $\pi^*_0 \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}^*_0, \mathsf{ct}^*_1, \mathbf{i}, \{z_{j,b}\}_{j\in[n],b\in\{0,1\}}), (\mathsf{sk}_0, \mathsf{sk}_1))$.

7. If for some $k \in \{1,2\}$, we have $\mathsf{R}_{1,0}((\mathsf{ct}_{0,k}, \mathsf{ct}_{1,k}, \mathbf{i}, \mathsf{pk}_0, \mathsf{pk}_1), (\mathsf{sk}_0, \mathsf{sk}_1)) = 0$, then:
   (a) Extract $\mathsf{wit}_y = \mathsf{NIZK.Ext}(\mathsf{t}_{\mathsf{ext},1}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_{0,1}, \mathsf{ct}_{1,1}, \mathbf{i}, y), \pi_{1,1})$.
   (b) If $\mathsf{R}_{\mathcal{L}}(y, \mathsf{wit}_y) = 0$, output $\perp$.
   (c) Else, generate $\pi^*_1 \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}_1, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}^*_0, \mathsf{ct}^*_1, \mathbf{i}, y), \mathsf{wit}_y)$.

8. Else, generate $\pi^*_1 \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}^*_0, \mathsf{ct}^*_1, \mathbf{i}, y), (\mathsf{sk}_0, \mathsf{sk}_1))$.

9. Output $(\mathsf{ct}^*_0, \mathsf{ct}^*_1, \mathbf{i}, \pi^*_0, \pi^*_1)$.
```

Figure 11: Circuit $\hat{\hat{\mathsf{C}}}_{\mathsf{Add}}$

Outer Hybrid 5. This hybrid is identical to outer hybrid 4 except that we switch the tuple (g_0, g_1, g_2, g_3) hardwired into the modified MMap circuits $\hat{\mathsf{C}}_{\mathsf{Add}}$, $\hat{\mathsf{C}}_{\mathsf{Inv}}$, $\hat{\mathsf{C}}_{\mathsf{Mult}}$ and $\hat{\mathsf{C}}_{\mathsf{ext}}$ from a uniformly random tuple of group elements to a uniformly random DDH tuple, albeit with the same base element g_0. More formally, we uniformly sample $\gamma_1, \gamma_2 \leftarrow \mathbb{Z}_q$, and set

$$(g_1, g_2, g_3) = (g_0^{\gamma_1}, g_0^{\gamma_2}, g_0^{\gamma_1 \cdot \gamma_2}).$$

Outer Hybrid 6. In this hybrid, we make the following alterations to the manner in which the MMap is generated:

1. We switch the element $z_{j,0}$ in the public parameters from a non-member to a member for $\mathcal{L}$, i.e., we now sample $z_{j,0} \leftarrow \mathcal{L}$, along with a (unique) membership-witness $\mathsf{wit}_{z_{j,0}}$. The element $z_{j,1}$ continues to be a member for $\mathcal{L}$, i.e., we continue to sample $z_{j,1} \leftarrow \mathcal{L}$, along with the (unique) membership-witness $\mathsf{wit}_{z_{j,1}}$.

2. We switch the circuits for addition, inversion and multiplication as follows:

$$\hat{\hat{\mathsf{C}}}_{\mathsf{Add}} \mapsto \hat{\mathsf{C}}_{\mathsf{Add}} \quad , \quad \hat{\hat{\mathsf{C}}}_{\mathsf{Inv}} \mapsto \hat{\mathsf{C}}_{\mathsf{Inv}},$$
$$\hat{\hat{\mathsf{C}}}_{\mathsf{Mult}} \mapsto \hat{\mathsf{C}}_{\mathsf{Mult}},$$

```
$\widehat{\mathsf{C}}_{\mathsf{Inv}}[\{\mathsf{sk}_b, \mathsf{pk}_b, \mathsf{crs}_b\}_{b\in\{0,1\}}, \mathsf{wit}_{z_{j,1}}, \mathsf{t}_{\mathsf{ext},1}, (g_0, g_1, g_2, g_3)](\mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, \pi_0, \pi_1)$:

1. Output $\perp$ if $\mathbf{i} > \vec{1}_n$.
2. Output $\perp$ if $\mathsf{NIZK.Verify}(\mathsf{crs}_0, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, \{z_{j,b}\}_{j\in[n],b\in\{0,1\}}), \pi_0) = 0$.
3. Output $\perp$ if $\mathsf{NIZK.Verify}(\mathsf{crs}_1, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, y), \pi_1) = 0$.
4. For $b \in \{0,1\}$, set: $\mathsf{ct}^*_b = \mathsf{FHE.Eval}(\mathsf{pk}_b, \mathsf{ct}_b, \mathsf{C}_{\mathsf{Inv},\mathrm{FHE}})$.
5. If $\mathsf{R}_{0,0}((\mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, \mathsf{pk}_0, \mathsf{pk}_1), (\mathsf{sk}_0, \mathsf{sk}_1)) = 0$, then:
   (a) // Omitted, depends on $\mathsf{t}_{\mathsf{ext},0}$.
   (b) Generate $\pi^*_0 \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}_0, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}^*_0, \mathsf{ct}^*_1, \mathbf{i}, \{z_{j,b}\}_{j\in[n],b\in\{0,1\}}), (j, \mathsf{wit}_{z_{j,1}}))$.
6. Else, generate $\pi^*_0 \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}^*_0, \mathsf{ct}^*_1, \mathbf{i}, \{z_{j,b}\}_{j\in[n],b\in\{0,1\}}), (\mathsf{sk}_0, \mathsf{sk}_1))$.
7. If $\mathsf{R}_{1,0}((\mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, \mathsf{pk}_0, \mathsf{pk}_1), (\mathsf{sk}_0, \mathsf{sk}_1)) = 0$, then:
   (a) Extract $\mathsf{wit}_y = \mathsf{NIZK.Ext}(\mathsf{t}_{\mathsf{ext},1}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, y), \pi_1)$.
   (b) If $\mathsf{R}_{\mathcal{L}}(y, \mathsf{wit}_y) = 0$, output $\perp$.
   (c) Else, generate $\pi^*_1 \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}_1, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}^*_0, \mathsf{ct}^*_1, \mathbf{i}, y), \mathsf{wit}_y)$.
8. Else, generate $\pi^*_1 \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}^*_0, \mathsf{ct}^*_1, \mathbf{i}, y), (\mathsf{sk}_0, \mathsf{sk}_1))$.
9. Output $(\mathsf{ct}^*_0, \mathsf{ct}^*_1, \mathbf{i}, \pi^*_0, \pi^*_1)$.
```

Figure 12: Circuit $\widehat{\widehat{\mathsf{C}}}_{\mathsf{Inv}}$

```
$\widehat{\widehat{\mathsf{C}}}_{\mathsf{Mult,FHE}}((\{a_{\ell,1}\}_{\ell\in[0,3]}, \mathbf{i}_1), (\{a_{\ell,2}\}_{\ell\in[0,3]}, \mathbf{i}_2), (\mathsf{flag}_1, \mathsf{flag}_2, 0, 0, \vec{0}_n)))$:

1. If $(a_{1,1}, a_{2,1}, a_{3,1}) = (0,0,0)$ and $\mathsf{flag}_2 = 1$, then output $((\{a_{0,1} \cdot a_{\ell,2}\}_{\ell\in[0,3]}, (\mathbf{i}_1 + \mathbf{i}_2)))$.
2. Else if $(a_{1,2}, a_{2,2}, a_{3,2}) = (0,0,0)$ and $\mathsf{flag}_1 = 1$, then output $((\{a_{0,2} \cdot a_{\ell,1}\}_{\ell\in[0,3]}, (\mathbf{i}_1 + \mathbf{i}_2)))$.
3. Else output $\perp$.
   // Omitted use of the secret exponents $(\gamma_1, \gamma_2, \gamma_3)$.
```

Figure 13: Circuits $\widehat{\widehat{\mathsf{C}}}_{\mathsf{Mult,FHE}}$

Outer Hybrid 7. This hybrid is identical to the outer hybrid 6, except that the challenge SXDH encodings are now switched back to a partially oblique form. In particular, the second FHE ciphertext in each encoding now encrypts a normal representation of the underlying plaintext element. Each encoding continues to have a NIZK proof π_0 for $z_{j,1} \in \mathcal{L}$ under the binding crs_0, and a NIZK proof π_1 for consistency with respect to extraction under the binding crs_1. For ease of understanding, we explicitly lay out what the FHE ciphertexts in each of the encodings actually encrypt in Table 6. The changes from outer hybrid 6 are highlighted in red.

Outer Hybrid 8. This hybrid is identical to the outer hybrid 7, except that the challenge SXDH encodings are now switched back entirely to the normal form. In particular, the first FHE ciphertext in each encoding now also encrypts a normal representation of the underlying plaintext element. Each encoding continues to have a NIZK proof π_0 for $z_{j,1} \in \mathcal{L}$ under the binding crs_0, and a NIZK proof π_1 for consistency with respect to extraction under the binding crs_1.

$\widehat{\widehat{\mathsf{C}}}_{\mathsf{Mult}}[\{\mathsf{sk}_b, \mathsf{pk}_b, \mathsf{crs}_b\}_{b\in\{0,1\}}, \mathsf{wit}_{z_{j,1}}, \mathsf{t}_{\mathsf{ext},1}, (g_0, \gamma_1, \gamma_2, \gamma_3)]\left(\{(\mathsf{ct}_{0,k}, \mathsf{ct}_{1,k}, \mathbf{i}_k, \pi_{0,k}, \pi_{1,k})\}_{k\in\{1,2\}}\right)$:

1. Output $\perp$ if $\mathbf{i}_1 + \mathbf{i}_2 > \vec{1}_n$.
2. Output $\perp$ if for any $k \in \{1,2\}$, we have
$$\mathsf{NIZK.Verify}(\mathsf{crs}_0, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_{0,k}, \mathsf{ct}_{1,k}, \mathbf{i}_k, \{z_{j,b}\}_{j\in[n],b\in\{0,1\}}), \pi_{0,k}) = 0.$$
3. Output $\perp$ if for any $k \in \{1,2\}$, $\mathsf{NIZK.Verify}(\mathsf{crs}_1, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_{0,k}, \mathsf{ct}_{1,k}, \mathbf{i}_k, y), \pi_{1,k}) = 0$.
4. For $k \in \{1,2\}$ set
$$\mathsf{flag}_k = \begin{cases} 0 & \text{if } \mathbf{i}_{k,j} = 0, \\ 1 & \text{otherwise.} \end{cases}$$
// Omitted use of $\mathsf{t}_{\mathsf{ext},0}$ above.
5. For $b \in \{0,1\}$, set: $\mathsf{ct}^*_b = \mathsf{FHE.Eval}(\mathsf{pk}_b, \mathsf{ct}_{b,1}, \mathsf{ct}_{b,2}, \mathsf{ct}_{b,\mathsf{flag}}, \widehat{\widehat{\mathsf{C}}}_{\mathsf{Mult,FHE}})$, where
$$\mathsf{ct}_{b,\mathsf{flag}} = \mathsf{FHE.Enc}(\mathsf{pk}_b, (\mathsf{flag}_1, \mathsf{flag}_2, 0, \vec{0}_n)).$$
// Omitted use of the secret exponents $(\gamma_1, \gamma_2, \gamma_3)$ above.
6. **If** there exists some $k \in 1, 2$ such that:
$$\mathsf{R}_{1,0}((\mathsf{ct}_{0,k}, \mathsf{ct}_{1,k}, \mathbf{i}_k, \mathsf{pk}_0, \mathsf{pk}_1), (\mathsf{sk}_0, \mathsf{sk}_1)) = 0 \text{ and } \mathbf{i}_{k,j} = 1,$$
then generate
$$\pi^*_0 \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}^*_0, \mathsf{ct}^*_1, (\mathbf{i}_1 + \mathbf{i}_2), \{z_{j,b}\}_{j\in[n],b\in\{0,1\}}), (j_k, \mathsf{wit}_{z,k})).$$
7. **Else**, generate
$$\pi^*_0 \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}^*_0, \mathsf{ct}^*_1, (\mathbf{i}_1 + \mathbf{i}_2), \{z_{j,b}\}_{j\in[n],b\in\{0,1\}}), (\mathsf{sk}_0, \mathsf{sk}_1)).$$
8. **If** for some $k \in \{1,2\}$, we have $\mathsf{R}_{1,0}((\mathsf{ct}_{0,k}, \mathsf{ct}_{1,k}, \mathbf{i}_k, \mathsf{pk}_0, \mathsf{pk}_1), (\mathsf{sk}_0, \mathsf{sk}_1)) = 0$, then:
   (a) Extract $\mathsf{wit}_y = \mathsf{NIZK.Ext}(\mathsf{t}_{\mathsf{ext},1}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_{0,1}, \mathsf{ct}_{1,1}, \mathbf{i}_k, y), \pi_{1,k})$.
   (b) **If** $\mathsf{R}_{\mathcal{L}}(y, \mathsf{wit}_y) = 0$, output $\perp$.
   (c) **Else**, generate $\pi^*_1 \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}_1, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}^*_0, \mathsf{ct}^*_1, (\mathbf{i}_1 + \mathbf{i}_2), y), \mathsf{wit}_y)$.
9. **Else**, generate $\pi^*_1 \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}^*_0, \mathsf{ct}^*_1, (\mathbf{i}_1 + \mathbf{i}_2), y), (\mathsf{sk}_0, \mathsf{sk}_1))$.
10. Output $(\mathsf{ct}^*_0, \mathsf{ct}^*_1, \mathbf{i}, \pi^*_0, \pi^*_1)$.

Figure 14: Circuit $\widehat{\widehat{\mathsf{C}}}_{\mathsf{Mult}}$

For ease of understanding, we explicitly lay out what the FHE ciphertexts in each of the encodings actually encrypt in Table 7. The changes from outer hybrid 7 are highlighted in red.

Outer Hybrid 9. This hybrid is identical to outer hybrid 8, except that we make the following alterations to the manner in which the MMap is set up:

1. We switch back the elements $z_{j,0}$ and $z_{j,1}$ in the public parameters from members to non-members for $\mathcal{L}$, i.e., we now sample $z_{j,0}, z_{j,1} \leftarrow \mathcal{X} \setminus \mathcal{L}$. Note that this is exactly as in the real MMap scheme.

Encoding	ct_0 **encrypts**	ct_1 **encrypts**	**Witness for** π_0	**Witness for** π_1
$[\alpha_0]$	$(\mu, 0, 0, 0, \mathbf{i})$	$(\mu, 0, 0, 0, \mathbf{i})$	$(j, \mathsf{wit}_{z_{j,1}})$	$(\mathsf{sk}_0, \mathsf{sk}_1)$
$[\alpha_1]$	$(0, \mu, 0, 0, \mathbf{i})$	$(\mu \cdot \gamma_1, 0, 0, 0, \mathbf{i})$	$(j, \mathsf{wit}_{z_{j,1}})$	$(\mathsf{sk}_0, \mathsf{sk}_1)$
$[\alpha_2]$	$(0, 0, \mu, 0, \mathbf{i})$	$(\mu \cdot \gamma_2, 0, 0, 0, \mathbf{i})$	$(j, \mathsf{wit}_{z_{j,1}})$	$(\mathsf{sk}_0, \mathsf{sk}_1)$
$[\alpha_3]$	$(0, 0, 0, \mu, \mathbf{i})$	$(\mu \cdot \gamma_1 \cdot \gamma_2, 0, 0, 0, \mathbf{i})$	$(j, \mathsf{wit}_{z_{j,1}})$	$(\mathsf{sk}_0, \mathsf{sk}_1)$

Table 6: Overview of Challenge Encodings in Outer Hybrid 7

Encoding	ct_0 **encrypts**	ct_1 **encrypts**	**Witness for** π_0	**Witness for** π_1
$[\alpha_0]$	$(\mu, 0, 0, 0, \mathbf{i})$	$(\mu, 0, 0, 0, \mathbf{i})$	$(j, \mathsf{wit}_{z_{j,1}})$	$(\mathsf{sk}_0, \mathsf{sk}_1)$
$[\alpha_1]$	$(\mu \cdot \gamma_1, 0, 0, 0, \mathbf{i})$	$(\mu \cdot \gamma_1, 0, 0, 0, \mathbf{i})$	$(j, \mathsf{wit}_{z_{j,1}})$	$(\mathsf{sk}_0, \mathsf{sk}_1)$
$[\alpha_2]$	$(\mu \cdot \gamma_2, 0, 0, 0, \mathbf{i})$	$(\mu \cdot \gamma_2, 0, 0, 0, \mathbf{i})$	$(j, \mathsf{wit}_{z_{j,1}})$	$(\mathsf{sk}_0, \mathsf{sk}_1)$
$[\alpha_3]$	$(\mu \cdot \gamma_1 \cdot \gamma_2, 0, 0, 0, \mathbf{i})$	$(\mu \cdot \gamma_1 \cdot \gamma_2, 0, 0, 0, \mathbf{i})$	$(j, \mathsf{wit}_{z_{j,1}})$	$(\mathsf{sk}_0, \mathsf{sk}_1)$

Table 7: Overview of Challenge Encodings in Outer Hybrid 8

2. We switch back the circuits for addition, inversion, multiplication and extraction as follows:

$$\begin{array}{ccc} \widehat{\mathsf{C}}_{\mathsf{Add}} \mapsto \mathsf{C}_{\mathsf{Add}} & , & \widehat{\mathsf{C}}_{\mathsf{Inv}} \mapsto \mathsf{C}_{\mathsf{Inv}}, \\ \widehat{\mathsf{C}}_{\mathsf{Mult}} \mapsto \mathsf{C}_{\mathsf{Mult}} & , & \widehat{\mathsf{C}}_{\mathsf{ext}} \mapsto \mathsf{C}_{\mathsf{ext}}, \end{array}$$

   In particular, the circuits are now exactly as in the real MMap scheme, with the exception that the tuple (g_0, g_1, g_2, g_3) hardwired inside these circuits continues to be a DDH tuple (and the corresponding secret exponents hardwired into the multiplication circuit are γ_1, γ_2 and $\gamma_1 \cdot \gamma_2$).

Additionally, we make the following change to the manner in which the challenge encodings are generated: for each encoding, the NIZK proof π_0 under the binding crs_0 now proves that the encoding is in the normal representation using the witness $(\mathsf{sk}_0, \mathsf{sk}_1)$, as opposed to proving that $z_{j,1} \in \mathcal{L}$. For ease of understanding, we explicitly lay out what the FHE ciphertexts in each of the encodings actually encrypt in Table 8. The changes from outer hybrid 8 are highlighted in red.

Outer Hybrid 10. This hybrid is identical to outer hybrid 7 except that we switch the tuple of group elements (g_0, g_1, g_2, g_3) hardwired into the MMap circuits $\mathsf{C}_{\mathsf{Add}}$, $\mathsf{C}_{\mathsf{Inv}}$, $\mathsf{C}_{\mathsf{Mult}}$ and $\mathsf{C}_{\mathsf{ext}}$ from a uniform DDH tuple to a uniformly random tuple, albeit with the same base element g_0 (and the corresponding secret exponents hardwired into the multiplication circuit are switched from $(\gamma_1, \gamma_2, \gamma_1 \cdot \gamma_2)$ to uniformly random $(\gamma_1, \gamma_2, \gamma 3)$). In other words, the MMap circuits are now exactly as in the real scheme.

Encoding	ct_0 **encrypts**	ct_1 **encrypts**	**Witness for** π_0	**Witness for** π_1
$[\alpha_0]$	$(\mu, 0, 0, 0, \mathbf{i})$	$(\mu, 0, 0, 0, \mathbf{i})$	$(\mathsf{sk}_0, \mathsf{sk}_1)$	$(\mathsf{sk}_0, \mathsf{sk}_1)$
$[\alpha_1]$	$(\mu \cdot \gamma_1, 0, 0, 0, \mathbf{i})$	$(\mu \cdot \gamma_1, 0, 0, 0, \mathbf{i})$	$(\mathsf{sk}_0, \mathsf{sk}_1)$	$(\mathsf{sk}_0, \mathsf{sk}_1)$
$[\alpha_2]$	$(\mu \cdot \gamma_2, 0, 0, 0, \mathbf{i})$	$(\mu \cdot \gamma_2, 0, 0, 0, \mathbf{i})$	$(\mathsf{sk}_0, \mathsf{sk}_1)$	$(\mathsf{sk}_0, \mathsf{sk}_1)$
$[\alpha_3]$	$(\mu \cdot \gamma_1 \cdot \gamma_2, 0, 0, 0, \mathbf{i})$	$(\mu \cdot \gamma_1 \cdot \gamma_2, 0, 0, 0, \mathbf{i})$	$(\mathsf{sk}_0, \mathsf{sk}_1)$	$(\mathsf{sk}_0, \mathsf{sk}_1)$

Table 8: Overview of Challenge Encodings in Outer Hybrid 9

6.2 Indistinguishability of Outer Hybrids

In this section, we argue that the outer hybrids are computationally indistinguishable from each other. Each argument in turn involves a sequence of inner hybrids, as described below.

Outer Hybrids 0 and 1. We first argue that outer hybrids 0 and 1 are computationally indistinguishable from each other. In particular, we state and prove the following lemma:

Lemma 6.2. *The outer hybrids 0 and 1 are computationally indistinguishable provided that all of the following assumptions hold:*

- *The piO scheme is indistinguishability-secure against X-IND samplers.*
- *Membership in $\mathcal{L}$ is computationally hard to decide,*
- *Members of $\mathcal{L}$ have unique witnesses for membership.*
- *The dual-mode NIZK proof system satisfies CRS indistinguishability, perfect extractability in the binding mode, and perfect witness-indistinguishability in the hiding mode.*

Proof. To prove this lemma, we use a sequence of inner hybrids. The first of these inner hybrids is identical to outer hybrid 0, while the last is identical to outer hybrid 1. Table 9 provides an overview of these inner hybrids.

Inner Hybrid 0-0. This hybrid is identical to outer hybrid 0.

Inner Hybrid 0-1. In this hybrid, we switch $z_{j,0}$ and $z_{j,1}$ in the public parameter of the MMap from uniform non-members to uniform members of $\mathcal{L}$. By the hardness of deciding membership in the set $\mathcal{L}$, inner hybrid 0-1 is computationally indistinguishable from inner hybrid 0-0 (more formally, we require two sub-hybrids - one each for switching each of the elements $z_{j,0}$ and $z_{j,1}$ from uniform non-members to uniform members of $\mathcal{L}$; we avoid this for brevity).

Inner Hybrid 0-2. In this hybrid we switch to using the modified MMap circuits $\widehat{\mathsf{C}}_{\mathsf{Add}}$, $\widehat{\mathsf{C}}_{\mathsf{Inv}}$, $\widehat{\mathsf{C}}_{\mathsf{Mult}}$ and $\widehat{\mathsf{C}}_{\mathsf{ext}}$. In particular, we hardwire the witnesses $\mathsf{wit}_{z_{j,0}}$ and $\mathsf{wit}_{z_{j,1}}$ for the membership of $z_{j,0}$ and $z_{j,1}$ into the circuits, and remove the extraction trapdoor $\mathsf{t}_{\mathsf{ext},0}$ from all of the circuits. We claim that this change does not change the functionality of the MMap circuits at all.

To begin with, observe that in order to reach the extraction step, the input encoding(s) to each circuit must pass the validity checks for π_0. Since crs_0 is binding in both inner hybrids 0-1 and 0-2, any valid encoding that passes this test must either contain a verifying proof of normal representation, or must contain a verifying proof of language-membership for $z_{j,0}$ or $z_{j,1}$. Now observe the following:

Inner Hybrid	crs_0	MMap Circuits	$(z_{j,0}, z_{j,1})$	Challenge Encodings				Remark
				SXDH/Random	Representation	Witness for π_0	Witness for π_1	
0-0	Binding	C_{op}	$z_{j,0}, z_{j,1} \notin \mathcal{L}$	Random	Normal	$(\mathsf{sk}_0, \mathsf{sk}_1)$	$(\mathsf{sk}_0, \mathsf{sk}_1)$	
0-1	Binding	C_{op}	$z_{j,0}, z_{j,1} \in \mathcal{L}$	Random	Normal	$(\mathsf{sk}_0, \mathsf{sk}_1)$	$(\mathsf{sk}_0, \mathsf{sk}_1)$	$\mathcal{L}$-hardness
0-2	Binding	$\widehat{\mathsf{C}}_{\mathsf{op}}$	$z_{j,0}, z_{j,1} \in \mathcal{L}$	Random	Normal	$(\mathsf{sk}_0, \mathsf{sk}_1)$	$(\mathsf{sk}_0, \mathsf{sk}_1)$	piO-security + unique $\mathsf{wit}_{z_{j,0}}$
0-3	Hiding	$\widehat{\mathsf{C}}_{\mathsf{op}}$	$z_{j,0}, z_{j,1} \in \mathcal{L}$	Random	Normal	$(\mathsf{sk}_0, \mathsf{sk}_1)$	$(\mathsf{sk}_0, \mathsf{sk}_1)$	NIZK CRS-indistinguishability
0-4	Hiding	$\widehat{\mathsf{C}}_{\mathsf{op}}$	$z_{j,0}, z_{j,1} \in \mathcal{L}$	Random	Normal	$(j, \mathsf{wit}_{z_{j,1}})$	$(\mathsf{sk}_0, \mathsf{sk}_1)$	NIZK witness-indistinguishability
0-5	Binding	$\widehat{\mathsf{C}}_{\mathsf{op}}$	$z_{j,0}, z_{j,1} \in \mathcal{L}$	Random	Normal	$(j, \mathsf{wit}_{z_{j,1}})$	$(\mathsf{sk}_0, \mathsf{sk}_1)$	NIZK CRS-indistinguishability

Table 9: Overview of the inner hybrids in the proof of indistinguishability of outer hybrids 0 and 1. Changes between subsequent hybrids are highlighted in red. Throughout, crs_1 is binding, $z_{j,1}, y \notin \mathcal{L}$, (g_0, g_1, g_2, g_3) is a random tuple of group elements, and the challenge encodings are both in normal representation as well as consistent with respect to extraction. We use the shorthands C_{op} and $\widehat{\mathsf{C}}_{\mathsf{op}}$ for the tuples $(\mathsf{C}_{\mathsf{Add}}, \mathsf{C}_{\mathsf{Inv}}, \mathsf{C}_{\mathsf{Mult}}, \mathsf{C}_{\mathsf{ext}})$ and $(\widehat{\mathsf{C}}_{\mathsf{Add}}, \widehat{\mathsf{C}}_{\mathsf{Inv}}, \widehat{\mathsf{C}}_{\mathsf{Mult}}, \widehat{\mathsf{C}}_{\mathsf{ext}})$, respectively, where the second set of circuits are described in detail subsequently.

- In the former case, the first **If** condition in the addition and inversion circuits, the **Else If** conditions in the multiplication circuit, and the first **If** condition in the extraction circuit are never satisfied, and the proof π_0^* for the output encoding is generated using the witness $(\mathsf{sk}_0, \mathsf{sk}_1)$. Hence, in this case, the outputs of the original and modified MMap circuits are identical.

- In the latter case, these conditions may be satisfied (if one or more of the input encodings to each circuit are in the oblique representation). In this case, to generate the proof π_0^* for the output encoding, the circuits $\mathsf{C}_{\mathsf{Add}}$, $\mathsf{C}_{\mathsf{Inv}}$ and $\mathsf{C}_{\mathsf{Mult}}$ use the extracted witness, while the circuits $\widehat{\mathsf{C}}_{\mathsf{Add}}$, $\widehat{\mathsf{C}}_{\mathsf{Inv}}$ and $\widehat{\mathsf{C}}_{\mathsf{Mult}}$ use the hardwired witness. Since the NIZK system has perfect extractability in the binding mode, extraction always succeeds in the original MMap circuits. Since $\mathcal{L}$ has unique membership witnesses, the extracted and hardwired witnesses must be identical. Hence, even in this case, the outputs of the original and modified MMap circuits are identical.

At this point, the transition can be justified by invoking the security of the piO scheme against X-IND samplers (once for each MMap circuit).

Inner Hybrid 0-3. In this hybrid we switch the string crs_0 in the public parameter from binding to hiding mode. Hence proofs generated under crs_0 will be perfectly witness indistinguishable in this hybrid. This hop can be justified by the CRS indistinguishability of the dual-mode NIZK proof system, and the fact that the modified MMap circuits $\widehat{\mathsf{C}}_{\mathsf{Add}}$, $\widehat{\mathsf{C}}_{\mathsf{Inv}}$, $\widehat{\mathsf{C}}_{\mathsf{Mult}}$ and $\widehat{\mathsf{C}}_{\mathsf{ext}}$ do not use the extraction trapdoor $\mathsf{t}_{\mathsf{ext},0}$ any longer.

Inner Hybrid 0-4. In this hybrid we switch the proof π_0 in each challenge encoding from using the witness $(\mathsf{sk}_0, \mathsf{sk}_1)$ to prove normal representation to using the witness $(j, \mathsf{wit}_{z_{j,1}})$ for proving the membership of $z_{j,1}$ in $\mathcal{L}$. Note that we still generate proofs that are valid, albeit using a different witness. This hop can be justified by the perfect witness-indistinguishability of the NIZK proof system in the hiding mode.

Inner Hybrid 0-5. In this hybrid we switch the string crs_0 in the public parameter back to binding mode from hiding mode. This hop can be justified by the CRS indistinguishability of the dual-mode NIZK proof system, and the fact that all proofs in the challenge encodings are valid.

Observe that in inner hybrid 0-5, the public parameters of the MMap and the challenge encodings are distributed exactly as in outer hybrid 1. This concludes the proof of Lemma 6.2. □

Inner Hybrid	$(\mathsf{crs}_0, \mathsf{crs}_1)$	MMap Circuits	y	Challenge Encodings				Remark
				SXDH/Random	Representation	Witness for π_0	Witness for π_1	
1-0	(Binding, Binding)	$\widehat{\mathsf{C}}_{\mathsf{op}}$	$y \notin \mathcal{L}$	Random	Normal	$(j, \mathsf{wit}_{z_{j,1}})$	$(\mathsf{sk}_0, \mathsf{sk}_1)$	
1-1	(Binding, Hiding)	$\widetilde{\mathsf{C}}_{\mathsf{op},0}$	$y \in \mathcal{L}$	Random	Normal	$(j, \mathsf{wit}_{z_{j,1}})$	wit_y	Lemma 6.4
1-2	(Biding, Hiding)	$\widetilde{\mathsf{C}}_{\mathsf{op},0}$	$y \in \mathcal{L}$	Random	Partially oblique	$(j, \mathsf{wit}_{z_{j,1}})$	wit_y	FHE CPA-security
1-3	(Binding, Binding)	$\widehat{\mathsf{C}}_{\mathsf{op}}$	$y \notin \mathcal{L}$	Random	Partially oblique	$(j, \mathsf{wit}_{z_{j,1}})$	$(\mathsf{sk}_0, \mathsf{sk}_1)$	Lemma 6.5

Table 10: Overview of the inner hybrids in the proof of indistinguishability of outer hybrids 1 and 2. Changes between subsequent hybrids are highlighted in red. Throughout, crs_0 is in binding mode, $z_{j,0}, z_{j,1} \in \mathcal{L}$, and (g_0, g_1, g_2, g_3) is a random tuple of group elements. We use the shorthands $\widehat{\mathsf{C}}_{\mathsf{op}}$ and $\widetilde{\mathsf{C}}_{\mathsf{op},0}$ for the tuples $(\widehat{\mathsf{C}}_{\mathsf{Add}}, \widehat{\mathsf{C}}_{\mathsf{Inv}}, \widehat{\mathsf{C}}_{\mathsf{Mult}}, \widehat{\mathsf{C}}_{\mathsf{ext}})$ and $(\widetilde{\mathsf{C}}_{\mathsf{Add},0}, \widetilde{\mathsf{C}}_{\mathsf{Inv},0}, \widetilde{\mathsf{C}}_{\mathsf{Mult},0}, \widetilde{\mathsf{C}}_{\mathsf{ext},0})$, respectively, where the second set of circuits are described in detail subsequently.

Outer Hybrids 1 and 2. We now argue that outer hybrids 1 and 2 are computationally indistinguishable from each other. In particular, we state and prove the following lemma:

Lemma 6.3. *The outer hybrids 1 and 2 are computationally indistinguishable provided that all of the following assumptions hold:*

- *The piO scheme is indistinguishability-secure against X-IND samplers.*
- *Membership in $\mathcal{L}$ is computationally hard to decide,*
- *Members of $\mathcal{L}$ have unique witnesses for membership.*
- *The dual-mode NIZK proof system satisfies CRS indistinguishability, perfect extractability in the binding mode, and perfect witness-indistinguishability in the hiding mode.*

Proof. To prove this lemma, we use another sequence of inner hybrids. The first of these inner hybrids is identical to outer hybrid 1, while the last is identical to outer hybrid 2. Table 10 provides an overview of these inner hybrids.

Inner Hybrid 1-0. This hybrid is identical to outer hybrid 1.

Inner Hybrid 1-1. In this hybrid, we make the following alterations:

1. We switch the string crs_1 from binding to hiding mode.
2. We switch the element y in the public parameters from a non-member to a member for $\mathcal{L}$, i.e., we now sample $y \leftarrow \mathcal{L}$, along with a (unique) membership-witness wit_y.
3. We switch the circuits for addition, inversion, multiplication and extraction as follows:

$$\begin{aligned} \widehat{\mathsf{C}}_{\mathsf{Add}} &\mapsto \widetilde{\mathsf{C}}_{\mathsf{Add}} \quad , \quad \widehat{\mathsf{C}}_{\mathsf{Inv}} \mapsto \widetilde{\mathsf{C}}_{\mathsf{Inv}}, \\ \widehat{\mathsf{C}}_{\mathsf{Mult}} &\mapsto \widetilde{\mathsf{C}}_{\mathsf{Mult}} \quad , \quad \widehat{\mathsf{C}}_{\mathsf{ext}} \mapsto \widetilde{\mathsf{C}}_{\mathsf{ext},0}. \end{aligned}$$

4. Additionally, we make the following change to the manner in which the challenge encodings are generated: for each encoding, the NIZK proof π_1 under crs_1 now proves that $y \in \mathcal{L}$ using the witness wit_y as opposed to proving consistency.

```
$\widetilde{\mathsf{C}}_{\mathsf{Add}}[\{\mathsf{sk}_b, \mathsf{pk}_b, \mathsf{crs}_b\}_{b\in\{0,1\}}, \{\mathsf{wit}_{z_{j,b}}\}_{b\in\{0,1\}}, \mathsf{wit}_y, (g_0, g_1, g_2, g_3)]\big(\{(\mathsf{ct}_{0,k}, \mathsf{ct}_{1,k}, \mathbf{i}_k, \pi_{0,k}, \pi_{1,k})\}_{k\in\{1,2\}}\big)$:

1. Output $\perp$ if $\mathbf{i}_1 \neq \mathbf{i}_2$ or $\mathbf{i}_1 > \mathbf{n}$. Else, set $\mathbf{i} = \mathbf{i}_1$ and proceed.
2. Output $\perp$ if for any $k \in \{1,2\}$, we have
       $\mathsf{NIZK.Verify}(\mathsf{crs}_0, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_{0,k}, \mathsf{ct}_{1,k}, \mathbf{i}, \{z_{j,b}\}_{j\in[n],b\in\{0,1\}}), \pi_{0,k}) = 0.$
3. Output $\perp$ if for any $k \in \{1,2\}$, we have
       $\mathsf{NIZK.Verify}(\mathsf{crs}_1, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_{0,k}, \mathsf{ct}_{1,k}, \mathbf{i}, y), \pi_{1,k}) = 0.$
4. For $b \in \{0,1\}$, set: $\mathsf{ct}^*_b = \mathsf{FHE.Eval}(\mathsf{pk}_b, \mathsf{ct}_{b,1}, \mathsf{ct}_{b,2}, \mathsf{C}_{\mathsf{Add},\mathrm{FHE}})$.
5. // Check omitted, depends on $(\mathsf{sk}_0, \mathsf{sk}_1)$.
6. Generate $\pi^*_0 \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}^*_0, \mathsf{ct}^*_1, \mathbf{i}, \{z_{j,b}\}_{j\in[n],b\in\{0,1\}}), (j, \mathsf{wit}_{z_{j,\mathbf{i}_j}}))$.
7. // Check omitted, depends on $(\mathsf{sk}_0, \mathsf{sk}_1)$.
8. Generate $\pi^*_1 \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}^*_0, \mathsf{ct}^*_1, \mathbf{i}, y), \mathsf{wit}_y)$.
9. Output $(\mathsf{ct}^*_0, \mathsf{ct}^*_1, \mathbf{i}, \pi^*_0, \pi^*_1)$.
```

Figure 15: Circuit $\widetilde{\mathsf{C}}_{\mathsf{Add}}$

```
$\widetilde{\mathsf{C}}_{\mathsf{Inv}}[\{\mathsf{sk}_b, \mathsf{pk}_b, \mathsf{crs}_b\}_{b\in\{0,1\}}, \{\mathsf{wit}_{z_{j,b}}\}_{b\in\{0,1\}}, \mathsf{wit}_y, (g_0, g_1, g_2, g_3)](\mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, \pi_0, \pi_1)$:

1. Output $\perp$ if $\mathbf{i} > \vec{1}_n$.
2. Output $\perp$ if $\mathsf{NIZK.Verify}(\mathsf{crs}_0, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, \{z_{j,b}\}_{j\in[n],b\in\{0,1\}}), \pi_0) = 0$.
3. Output $\perp$ if $\mathsf{NIZK.Verify}(\mathsf{crs}_1, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, y), \pi_1) = 0$.
4. For $b \in \{0,1\}$, set: $\mathsf{ct}^*_b = \mathsf{FHE.Eval}(\mathsf{pk}_b, \mathsf{ct}_b, \mathsf{C}_{\mathsf{Inv},\mathrm{FHE}})$.
5. // Check omitted, depends on $(\mathsf{sk}_0, \mathsf{sk}_1)$.
6. Generate $\pi^*_0 \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}^*_0, \mathsf{ct}^*_1, \mathbf{i}, \{z_{j,b}\}_{j\in[n],b\in\{0,1\}}), (j, \mathsf{wit}_{z_{j,\mathbf{i}_j}}))$.
7. // Check omitted, depends on $(\mathsf{sk}_0, \mathsf{sk}_1)$.
8. Generate $\pi^*_1 \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}^*_0, \mathsf{ct}^*_1, \mathbf{i}, y), \mathsf{wit}_y)$.
9. Output $(\mathsf{ct}^*_0, \mathsf{ct}^*_1, \mathbf{i}, \pi^*_0, \pi^*_1)$.
```

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

The modified circuits are described in Figures 15, 16, 17, and 18, respectively. At a high level, in each of these switched circuits, we hardwire the witness wit_y for the membership of y in $\mathcal{L}$ and avoid using the second extraction trapdoor $\mathsf{t}_{\mathsf{ext},1}$ inside the second **If** branch, which checks for consistency. In other words, we suspend all consistency checks in all MMap circuits.

Note that the modified circuits for addition, inversion and multiplication, namely $\widetilde{\mathsf{C}}_{\mathsf{Add}}$, $\widetilde{\mathsf{C}}_{\mathsf{Inv}}$ and $\widetilde{\mathsf{C}}_{\mathsf{Mult}}$, are hardwired with neither sk_0 nor sk_1. The modified extraction circuits $\widetilde{\mathsf{C}}_{\mathsf{ext},\beta}$ is only hardwired with $\mathsf{sk}_{1-\beta}$ and not sk_β.

$\widetilde{\mathsf{C}}_{\mathsf{Mult}}[\{\mathsf{sk}_b, \mathsf{pk}_b, \mathsf{crs}_b\}_{b\in\{0,1\}}, \{\mathsf{wit}_{z_{j,b}}\}_{b\in\{0,1\}}, \mathsf{wit}_y, (g_0, \gamma_1, \gamma_2, \gamma_3)]\left(\{(\mathsf{ct}_{0,k}, \mathsf{ct}_{1,k}, \mathbf{i}_k, \pi_{0,k}, \pi_{1,k})\}_{k\in\{1,2\}}\right)$:

1. Output $\perp$ if $\mathbf{i}_1 + \mathbf{i}_2 > \vec{1}_n$.
2. Output $\perp$ if for any $k \in \{1,2\}$, we have
$$\mathsf{NIZK.Verify}(\mathsf{crs}_0, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_{0,k}, \mathsf{ct}_{1,k}, \mathbf{i}_k, \{z_{j,b}\}_{j\in[n],b\in\{0,1\}}), \pi_{0,k}) = 0.$$
3. Output $\perp$ if for any $k \in \{1,2\}$, $\mathsf{NIZK.Verify}(\mathsf{crs}_1, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_{0,k}, \mathsf{ct}_{1,k}, \mathbf{i}_k, y), \pi_{1,k}) = 0$.
4. For $k \in \{1,2\}$ set
$$\mathsf{flag}_k = \begin{cases} 0 & \text{if } \mathbf{i}_{k,j} = 0, \\ 1 & \text{otherwise.} \end{cases}$$
// Omitted use of $\mathsf{t}_{\mathsf{ext},0}$ above.
5. For $b \in \{0,1\}$, set: $\mathsf{ct}^*_b = \mathsf{FHE.Eval}(\mathsf{pk}_b, \mathsf{ct}_{b,1}, \mathsf{ct}_{b,2}, \mathsf{ct}_{b,\gamma}, \mathsf{ct}_{b,\mathsf{flag}}, \mathsf{C}_{\mathsf{Mult},\mathrm{FHE}})$, where
$$\mathsf{ct}_{b,\gamma} = \mathsf{FHE.Enc}(\mathsf{pk}_b, (0, \gamma_1, \gamma_2, \gamma_3, \vec{0}_n)),$$
$$\mathsf{ct}_{b,\mathsf{flag}} = \mathsf{FHE.Enc}(\mathsf{pk}_b, (\mathsf{flag}_1, \mathsf{flag}_2, 0, \vec{0}_n)).$$
6. Letting $\mathbf{i}^* = \mathbf{i}_1 + \mathbf{i}_2$, generate:
$$\pi^*_0 \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}^*_0, \mathsf{ct}^*_1, \mathbf{i}^*, \{z_{j,b}\}_{j\in[n],b\in\{0,1\}}), (j, \mathsf{wit}_{z_{j,\mathbf{i}^*_j}})).$$
$$\pi^*_1 \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}^*_0, \mathsf{ct}^*_1, \mathbf{i}^*, y), \mathsf{wit}_y).$$
//All checks depending on $(\mathsf{sk}_0, \mathsf{sk}_1)$ omitted above.
7. Output $(\mathsf{ct}^*_0, \mathsf{ct}^*_1, \mathbf{i}, \pi^*_0, \pi^*_1)$.

Figure 17: Circuit $\widetilde{\mathsf{C}}_{\mathsf{Mult}}$

$\widetilde{\mathsf{C}}_{\mathsf{ext},\beta}[\mathsf{sk}_{1-\beta}, \{\mathsf{pk}_b, \mathsf{crs}_b\}_{b\in\{0,1\}}, (j, \mathsf{wit}_{z_{j,1}}), \mathsf{wit}_y, (g_0, g_1, g_2, g_3)](\mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, \pi_0, \pi_1)$ **for** $\beta \in \{0,1\}$**:**

1. Output $\perp$ if $\mathbf{i} > \vec{1}_n$.
2. Output $\perp$ if $\mathsf{NIZK.Verify}(\mathsf{crs}_0, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, \{z_{j,1}\}_{j\in[0,n]}), \pi_0) = 0$.
3. Output $\perp$ if $\mathsf{NIZK.Verify}(\mathsf{crs}_1, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, y), \pi_1) = 0$.
4. Recover $(\{a_{\ell,1-\beta}\}_{\ell\in[0,3]}, \mathbf{i}_{1-b}) = \mathsf{FHE.Dec}(\mathsf{sk}_{1-\beta}, \mathsf{ct}_{1-\beta})$.
5. Compute $g^* = \prod_{\ell\in[0,3]} g_\ell^{a_{\ell,1-\beta}}$.
6. // Check omitted, depends on sk_β.
7. // Check omitted, depends on sk_β.
8. Output g^*.

Figure 18: Circuit $\widetilde{\mathsf{C}}_{\mathsf{ext},\beta}$ for $\beta \in \{0,1\}$

More specifically, $\widetilde{\mathsf{C}}_{\mathsf{ext},0}$ is only hardwired with sk_1 and not sk_0.

We state and prove the following lemma:

Inner Hybrid	$(\mathsf{crs}_0, \mathsf{crs}_1)$	MMap Circuits		y	Challenge Encodings		Remark
		Overall Structure	Hardwired		Witness for π_0	Witness for π_1	
1-0-0	(Binding, Binding)	$\widehat{\mathsf{C}}_{\mathsf{op}}$	$(\mathsf{sk}_0, \mathsf{sk}_1, \mathsf{wit}_{z_{j,0}}, \mathsf{wit}_{z_{j,1}}, \mathsf{t}_{\mathsf{ext},1})$	$y \notin \mathcal{L}$	$(j, \mathsf{wit}_{z_{j,1}})$	$(\mathsf{sk}_0, \mathsf{sk}_1)$	
1-0-1	(Binding, Binding)	$\widehat{\mathsf{C}}_{\mathsf{op}}$	$(\mathsf{sk}_0, \mathsf{sk}_1, \mathsf{wit}_{z_{j,0}}, \mathsf{wit}_{z_{j,1}}, \mathsf{t}_{\mathsf{ext},1})$	$y \in \mathcal{L}$	$(j, \mathsf{wit}_{z_{j,1}})$	$(\mathsf{sk}_0, \mathsf{sk}_1)$	$\mathcal{L}$-hardness
1-0-2	(Binding, Binding)	$\widehat{\mathsf{C}}_{\mathsf{op}}$	$(\mathsf{sk}_0, \mathsf{sk}_1, \mathsf{wit}_{z_{j,0}}, \mathsf{wit}_{z_{j,1}}, \mathsf{wit}_y)$	$y \in \mathcal{L}$	$(j, \mathsf{wit}_{z_{j,1}})$	$(\mathsf{sk}_0, \mathsf{sk}_1)$	piO-security + unique wit_y
1-0-3	(Hiding, Binding)	$\widehat{\mathsf{C}}_{\mathsf{op}}$	$(\mathsf{sk}_0, \mathsf{sk}_1, \mathsf{wit}_{z_{j,0}}, \mathsf{wit}_{z_{j,1}}, \mathsf{wit}_y)$	$y \in \mathcal{L}$	$(j, \mathsf{wit}_{z_{j,1}})$	$(\mathsf{sk}_0, \mathsf{sk}_1)$	NIZK CRS-indistinguishability
1-0-4	(Hiding, Binding)	$\widehat{\mathsf{C}}_{\mathsf{op}}$ except Witnesses for output π_0^* are either $z_{j,0}$ or $z_{j,1}$	$(\mathsf{sk}_0, \mathsf{sk}_1, \mathsf{wit}_{z_{j,0}}, \mathsf{wit}_{z_{j,1}}, \mathsf{wit}_y)$	$y \in \mathcal{L}$	$(j, \mathsf{wit}_{z_{j,1}})$	$(\mathsf{sk}_0, \mathsf{sk}_1)$	piO-security + NIZK witness-indistinguishability
1-0-5	(Binding, Binding)	$\widehat{\mathsf{C}}_{\mathsf{op}}$ except Witnesses for output π_0^* are either $z_{j,0}$ or $z_{j,1}$	$(\mathsf{sk}_0, \mathsf{sk}_1, \mathsf{wit}_{z_{j,0}}, \mathsf{wit}_{z_{j,1}}, \mathsf{wit}_y)$	$y \in \mathcal{L}$	$(j, \mathsf{wit}_{z_{j,1}})$	$(\mathsf{sk}_0, \mathsf{sk}_1)$	NIZK CRS-indistinguishability
1-0-6	(Binding, Hiding)	$\widehat{\mathsf{C}}_{\mathsf{op}}$ except Witnesses for output π_0^* are either $z_{j,0}$ or $z_{j,1}$	$(\mathsf{sk}_0, \mathsf{sk}_1, \mathsf{wit}_{z_{j,0}}, \mathsf{wit}_{z_{j,1}}, \mathsf{wit}_y)$	$y \in \mathcal{L}$	$(j, \mathsf{wit}_{z_{j,1}})$	$(\mathsf{sk}_0, \mathsf{sk}_1)$	NIZK CRS-indistinguishability
1-0-7	(Binding, Hiding)	$\widetilde{\mathsf{C}}_{\mathsf{op},0}$	$(\mathsf{sk}_1, \mathsf{wit}_{z_{j,0}}, \mathsf{wit}_{z_{j,1}}, \mathsf{wit}_y)$	$y \in \mathcal{L}$	$(j, \mathsf{wit}_{z_{j,1}})$	$(\mathsf{sk}_0, \mathsf{sk}_1)$	piO-security + NIZK witness-indistinguishability
1-0-8	(Binding, Hiding)	$\widetilde{\mathsf{C}}_{\mathsf{op},0}$	$(\mathsf{sk}_1, \mathsf{wit}_{z_{j,0}}, \mathsf{wit}_y)$	$y \in \mathcal{L}$	$(j, \mathsf{wit}_{z_{j,1}})$	wit_y	NIZK witness-indistinguishability

Table 11: Overview of the hybrids in the proof of indistinguishability of inner hybrids 1-0 and 1-1. Changes between subsequent hybrids are highlighted in red. Throughout, $z_{j,0} \in \mathcal{L}$, and (g_0, g_1, g_2, g_3) is a random tuple of group elements. We use the shorthands $\widehat{\mathsf{C}}_{\mathsf{op}}$ and $\widetilde{\mathsf{C}}_{\mathsf{op},0}$ for the tuples $(\widehat{\mathsf{C}}_{\mathsf{Add}}, \widehat{\mathsf{C}}_{\mathsf{Inv}}, \widehat{\mathsf{C}}_{\mathsf{Mult}}, \widehat{\mathsf{C}}_{\mathsf{ext}})$ and $(\widetilde{\mathsf{C}}_{\mathsf{Add},0}, \widetilde{\mathsf{C}}_{\mathsf{Inv},0}, \widetilde{\mathsf{C}}_{\mathsf{Mult},0}, \widetilde{\mathsf{C}}_{\mathsf{ext},0})$, respectively, where the second set of circuits are described in detail subsequently.

Lemma 6.4. *The inner hybrids 1-0 and 1-1 are computationally indistinguishable provided that all of the following assumptions hold:*

- *The piO scheme is indistinguishability-secure against X-IND samplers.*
- *Membership in $\mathcal{L}$ is computationally hard to decide,*
- *Members of $\mathcal{L}$ have unique witnesses for membership.*
- *The dual-mode NIZK proof system satisfies CRS indistinguishability, perfect extractability in the binding mode, and perfect witness-indistinguishability in the hiding mode.*

Proof. To prove this lemma, we use an additional layer of inner hybrids. The first of these inner hybrids is identical to outer hybrid 1, while the last is identical to outer hybrid 2. Table 11 provides an overview of these inner hybrids. We would like to note here that these sequence of hybrids closely resemble those used by the authors of [FHHL18] in the proof of Lemma 6.2 in their paper.

1. **Inner Hybrid 1-0-0:** This hybrid is identical to inner hybrid 1-0.

2. **Inner Hybrid 1-0-1.** In this hybrid, we switch y in the public parameter of the MMap from a uniform non-member be a uniform member of $\mathcal{L}$. By the hardness of deciding membership in the set $\mathcal{L}$, inner hybrid 1-0-1 is computationally indistinguishable from inner hybrid 1-0-0.

3. **Inner Hybrid 1-0-2.** In this hybrid we take a first step towards switching to using the MMap circuits $\widetilde{\mathsf{C}}_{\mathsf{Add}}$, $\widetilde{\mathsf{C}}_{\mathsf{Inv}}$, $\widetilde{\mathsf{C}}_{\mathsf{Mult}}$ and $\widetilde{\mathsf{C}}_{\mathsf{ext}}$ as described in Figures 15, 16, 17 and 18, respectively. In particular, we make the following modification to the MMap circuits $\widehat{\mathsf{C}}_{\mathsf{Add}}$, $\widehat{\mathsf{C}}_{\mathsf{Inv}}$, $\widehat{\mathsf{C}}_{\mathsf{Mult}}$ and $\widehat{\mathsf{C}}_{\mathsf{ext}}$: we hardwire the witness wit_y for the membership of y into the circuits, and remove the extraction trapdoor $\mathsf{t}_{\mathsf{ext},1}$ from all of the circuits. We claim that this change

does not change the functionality of the MMap circuits at all. The proof of this claim is very similar to the proof of inner hybrid 0-2 (in the proof of Lemma 6.2).

To begin with, observe that in order to reach the extraction step, the input encoding(s) to each circuit must pass the validity checks for π_1. Since crs_1 is binding in both inner hybrids 1-0-1 and 1-0-2, any valid encoding that passes this test must either contain a verifying proof of consistency, or a verifying proof of language-membership for y. Now observe the following:

- In the former case, the second **If** condition in each MMap circuit is never satisfied, and the proof π_1^* for the output encoding is generated using the witness $(\mathsf{sk}_0, \mathsf{sk}_1)$. Hence, in this case, the outputs of the MMap circuits are identical across the inner hybrids 1-0-1 and 1-0-2.

- In the latter case, the second **If** condition in each MMap circuit may be satisfied (if the input encoding is inconsistent). In this case, to generate the proof π_1^* for the output encoding, the circuits in inner hybrid 1-0-1 use the extracted witness for the language-membership of y, while the circuits in the inner hybrid 1-0-2 use the hardwired witness for the language-membership of y. Since the NIZK system has perfect extractability in the binding mode, extraction always succeeds in the circuits in inner hybrid 1-0-1. Since $\mathcal{L}$ has unique membership witnesses, the extracted and hardwired witnesses must be identical. Hence, even in this case, the outputs of the MMap circuits are identical across the inner hybrids 1-0-1 and 1-0-2.

At this point, the transition can be justified by invoking the security of the piO scheme against X-IND samplers (once for each MMap circuit).

4. **Inner Hybrid 1-0-3.** In this hybrid we switch the strings crs_0 in the public parameter from binding to hiding mode. Hence proofs generated under crs_0 will be perfectly witness indistinguishable in this hybrid. This hop can be justified by the CRS indistinguishability of the dual-mode NIZK proof system, and the fact that the modified MMap circuits $\widehat{\mathsf{C}}_{\mathsf{Add}}$, $\widehat{\mathsf{C}}_{\mathsf{Inv}}$, $\widehat{\mathsf{C}}_{\mathsf{Mult}}$ and $\widehat{\mathsf{C}}_{\mathsf{ext}}$ do not use the extraction trapdoors $\mathsf{t}_{\mathsf{ext},0}$.

5. **Inner Hybrid 1-0-4.** In this hybrid we take a second step towards switching to using the MMap circuits $\widetilde{\mathsf{C}}_{\mathsf{Add}}$, $\widetilde{\mathsf{C}}_{\mathsf{Inv}}$, $\widetilde{\mathsf{C}}_{\mathsf{Mult}}$ and $\widetilde{\mathsf{C}}_{\mathsf{ext}}$ as described in Figures 15, 16, 17 and 18, respectively. In particular, we make the following modification: we *always* generate the proof π_0^* for the output encoding in the addition, inversion and multiplication circuits using either the hardwired witness $\mathsf{wit}_{z_{j,0}}$ for the language-membership of $z_{j,0}$ or the hardwired witness $\mathsf{wit}_{z_{j,1}}$ for the language-membership of $z_{j,1}$ (depending on the output level set), irrespective of whether the original encoding was in normal representation or oblique representation.

   We claim that this does not change the output distribution of the MMap circuits. To see this, observe that if the inputs to the MMap circuits are valid, then all proofs π_0^* generated by the circuits continue to remain valid. The *only* item that changes is the witness used to generate these proofs, irrespective of whether the original encoding was in normal representation or oblique representation. So there is a (potential) change of witnesses used to generate the proof π_0^* for the output encoding.

   However recall that crs_0 is in hiding mode in both inner hybrids 1-0-3 and 1-0-4; hence, by the perfect witness-indistinguishability of the NIZK proof system in the hiding mode, the distributions of these proofs, and hence the outputs of the MMap circuits, remain unaltered. Hence, this transition can be justified by invoking the security of the piO scheme against X-IND samplers (once for each of the three MMap circuits for addition, inversion and multiplication).

6. **Inner Hybrid 1-0-5.** In this hybrid we switch the string crs_0 in the public parameter back from hiding to binding mode. Hence proofs generated under crs_0 will be perfectly sound and extractable in this hybrid. This hop can be justified by the CRS indistinguishability of the dual-mode NIZK proof system, and the fact that the modified MMap circuits $\widehat{\mathsf{C}}_{\mathsf{Add}}$, $\widehat{\mathsf{C}}_{\mathsf{Inv}}$ and $\widehat{\mathsf{C}}_{\mathsf{Mult}}$ always produce valid proofs π_0^* for their output encodings in the inner hybrid 1-0-4 (and hence in this inner hybrid as well).

7. **Inner Hybrid 1-0-6.** In this hybrid we switch the string crs_1 in the public parameter from binding to hiding mode. Hence proofs generated under crs_1 will be perfectly witness indistinguishable in this hybrid. This hop can be justified by the CRS indistinguishability of the dual-mode NIZK proof system, and the fact that the modified MMap circuits $\widehat{\mathsf{C}}_{\mathsf{Add}}$, $\widehat{\mathsf{C}}_{\mathsf{Inv}}$, $\widehat{\mathsf{C}}_{\mathsf{Mult}}$ and $\widehat{\mathsf{C}}_{\mathsf{ext}}$ do not use the extraction trapdoor $\mathsf{t}_{\mathsf{ext},1}$ any longer.

8. **Inner Hybrid 1-0-7.** In this hybrid, we *completely* switch to using the MMap circuits $\widetilde{\mathsf{C}}_{\mathsf{Add}}$, $\widetilde{\mathsf{C}}_{\mathsf{Inv}}$, $\widetilde{\mathsf{C}}_{\mathsf{Mult}}$ and $\widetilde{\mathsf{C}}_{\mathsf{ext}}$ as described in Figures 15, 16, 17 and 18, respectively.

   Specifically, the *only* additional change from inner hybrid 1-0-6 is that we let the addition, inversion and multiplication circuits to always use the hardwired witness wit_y to generate the proof π_1^* in the output encodings, irrespective of whether the input encodings are consistent. Hence, these circuits need no longer perform the explicit checks for the relation R_1. This in turn means that they need neither sk_0 nor sk_1 any longer.

   We claim that these modifications do not change the output distributions of the MMap circuits. To see this, observe the following:

   - The string crs_0 is still in the binding mode, and hence any input encoding that passes the validity test for π_0 must either have a verifying proof of normal representation (and hence consistency), or a verifying proof for the language-membership of $z_{j,0}$. The witnesses used for these proofs have not changed from inner hybrid 1-0-6.
   - All proofs π_1^* generated in inner hybrid 1-0-6 are verifying, and hence all encodings output by the MMap circuits are valid. Indeed, we have *only* changed the witnesses used for the proof π_1^*. By the perfect witness-indistinguishability of the NIZK proof system in the hiding mode, the distributions of these proofs remain unaltered.

     Note that even in the original scheme, one could generate "valid" encodings that are "inconsistent" provided that they could produce a verifying proof of membership for y in $\mathcal{L}$. Hence, our suspension of consistency checks does not introduce any new functionality or change the existing functionality of the MMap circuits - it simply showcases how to use the OR branches in our NIZK statements as a proof construct. $\square$

   At this point, we can argue that the output distributions of the MMap circuits remain unaltered across inner hybrids 1-0-6 and 1-0-7, and the transition can be justified by invoking the security of the piO scheme against X-IND samplers (once for each MMap circuit).

9. **Inner Hybrid 1-0-8.** In this hybrid we switch the proof π_1 in each challenge encoding from using the witness $(\mathsf{sk}_0, \mathsf{sk}_1)$ to prove consistency to using the witness wit_y for proving the membership of y in $\mathcal{L}$. Note that we still generate proofs that are valid, and the *only* item that changes is the witness used to generate these proofs; hence, by the perfect witness-indistinguishability of the NIZK proof system in the hiding mode, the distributions of these proofs remain unaltered. This allows us to justify this final hop.

Observe that in inner hybrid 1-0-8, the public parameters of the MMap and the challenge encodings are distributed exactly as in inner hybrid 1-1. This concludes the proof of Lemma 6.4.

Inner Hybrid 1-2. In this hybrid, we switch the challenge encodings from normal representation to partially oblique representation. More specifically, we switch the plaintext tuples underlying the FHE ciphertexts in each challenge encoding as follows:

$$\begin{array}{llll}
[\alpha_1] : (\mu\cdot\gamma_1, 0, 0, 0, \mathbf{i}), (\mu\cdot\gamma_1, 0, 0, 0, \mathbf{i}) & \longrightarrow & (0, \mu, 0, 0, \mathbf{i}), (\mu\cdot\gamma_1, 0, 0, 0, \mathbf{i}) \\
[\alpha_2] : (\mu\cdot\gamma_2, 0, 0, 0, \mathbf{i}), (\mu\cdot\gamma_2, 0, 0, 0, \mathbf{i}) & \longrightarrow & (0, 0, \mu, 0, \mathbf{i}), (\mu\cdot\gamma_2, 0, 0, 0, \mathbf{i}) \\
[\alpha_3] : (\mu\cdot\gamma_3, 0, 0, 0, \mathbf{i}), (\mu\cdot\gamma_3, 0, 0, 0, \mathbf{i}) & \longrightarrow & (0, 0, 0, \mu, \mathbf{i}), (\mu\cdot\gamma_3, 0, 0, 0, \mathbf{i})
\end{array}$$

We argue that inner hybrid 1-1 is computationally indistinguishable from inner hybrid 1-2 assuming that FHE is CPA-secure. More specifically, we rely on multi-challenge FHE CPA-security, which is polynomially equivalent to single-challenge FHE CPA-security.

To see this, suppose that there exists a PPT adversary $\mathcal{A}$ that can distinguish between the inner hybrids 1-1 and 1-2 with non-negligible probability. We construct a PPT algorithm $\mathcal{B}$ that can break the multi-challenge CPA-security of the FHE scheme with non-negligible probability. From the challenger in the multi-challenge FHE CPA-security game, $\mathcal{B}$ receives a public key pk_0. It then sets up the public parameters for the MMap as follows:

1. $\mathcal{B}$ samples a different key-pair for the FHE scheme as:
$$(\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathsf{FHE.Gen}(1^\lambda),$$
$\mathcal{B}$ then samples $g_0 \leftarrow \mathbb{G}$ and $\gamma_1, \gamma_2, \gamma_3 \leftarrow \mathbb{Z}_q$, and sets:
$$g_1 = g_0^{\gamma_1} \quad , \quad g_2 = g_0^{\gamma_2} \quad , \quad g_2 = g_0^{\gamma_3}.$$
Finally, $\mathcal{B}$ samples a pair of NIZK CRS strings in the hiding mode as:
$$\mathsf{crs}_0, \mathsf{crs}_1 \leftarrow \mathsf{NIZK.Setup}(1^\lambda, \mathsf{hiding}).$$

2. Next $\mathcal{B}$ uniformly samples a total $(n+1)$ elements from the set $\mathcal{X}$ that are all non-members for the subset $\mathcal{L}$ and one element that is a member for $\mathcal{L}$. More formally, it samples
$$z_{1,0}, z_{1,1}, \ldots, z_{j-1,0}, z_{j-1,1}, z_{j+1,0}, z_{j+1,1} \ldots, z_{n,0}, z_{n,1} \leftarrow \mathcal{X} \setminus \mathcal{L},$$
$$y, z_{j,0}, z_{j,1} \leftarrow \mathcal{L},$$
where $z_{j,0}$ has unique membership-witness $\mathsf{wit}_{z_{j,0}}$, $z_{j,1}$ has unique membership-witness $\mathsf{wit}_{z_{j,1}}$, and y has unique membership-witness wit_y.

3. Finally, $\mathcal{B}$ sets up the MMap circuits $\widetilde{\mathsf{C}}_{\mathsf{Add},0}$, $\widetilde{\mathsf{C}}_{\mathsf{Inv},0}$, $\widetilde{\mathsf{C}}_{\mathsf{Mult},0}$ and $\widetilde{\mathsf{C}}_{\mathsf{ext},0}$ exactly as described in Figures 15, 16, 17 and 18, respectively. Note that the extraction circuit is only hardwired with sk_1 and not sk_0, while the remaining circuits are hardwired with neither sk_0 nor sk_1. Hence $\mathcal{B}$ does not require the knowledge of sk_0 to create these circuits. It then computes and outputs four probabilistically indistinguishability-obfuscated circuits of the form
$$\bar{\mathsf{C}}_{\mathsf{Add}} = \mathsf{piO.Obf}(\widetilde{\mathsf{C}}_{\mathsf{Add},0}) \quad , \quad \bar{\mathsf{C}}_{\mathsf{Inv}} = \mathsf{piO.Obf}(\widetilde{\mathsf{C}}_{\mathsf{Inv},0}),$$
$$\bar{\mathsf{C}}_{\mathsf{Mult}} = \mathsf{piO.Obf}(\widetilde{\mathsf{C}}_{\mathsf{Mult},0}) \quad , \quad \bar{\mathsf{C}}_{\mathsf{ext}} = \mathsf{piO.Obf}(\widetilde{\mathsf{C}}_{\mathsf{ext},0}).$$

Next, $\mathcal{B}$ sets up the challenge encodings as follows:

1. $\mathcal{B}$ samples $\mu \leftarrow \mathbb{Z}_q$ and provides the following pairs of challenge plaintexts to the challenger in the multi-challenge FHE CPA-security game:
$$\begin{aligned}
\text{Pair-1:} &\quad (\mu \cdot \gamma_1, 0, 0, 0, \mathbf{i}) \quad , \quad (0, \mu, 0, 0, \mathbf{i}),\\
\text{Pair-2:} &\quad (\mu \cdot \gamma_2, 0, 0, 0, \mathbf{i}) \quad , \quad (0, 0, \mu, 0, \mathbf{i}),\\
\text{Pair-3:} &\quad (\mu \cdot \gamma_3, 0, 0, 0, \mathbf{i}) \quad , \quad (0, 0, 0, \mu, \mathbf{i}).
\end{aligned}$$
In response, it receives from the challenger a tuple of ciphertexts $(\mathsf{ct}_1^*, \mathsf{ct}_2^*, \mathsf{ct}_3^*)$ under pk_0 and sets:
$$\mathsf{ct}_{0,0} = \mathsf{FHE.Enc}(\mathsf{pk}_0, (\mu, 0, 0, 0, \mathbf{i})),$$
$$\mathsf{ct}_{0,1} = \mathsf{ct}_1^* \quad , \quad \mathsf{ct}_{0,2} = \mathsf{ct}_2^* \quad , \quad \mathsf{ct}_{0,3} = \mathsf{ct}_3^*.$$

2. $\mathcal{B}$ then generates the following FHE ciphertexts:

$$\mathsf{ct}_{1,0} = \mathsf{FHE.Enc}(\mathsf{pk}_1, (\mu, 0, 0, 0, \mathbf{i})) \quad , \quad \mathsf{ct}_{1,1} = \mathsf{FHE.Enc}(\mathsf{pk}_1, (\mu \cdot \gamma_1, 0, 0, 0, \mathbf{i})),$$
$$\mathsf{ct}_{1,2} = \mathsf{FHE.Enc}(\mathsf{pk}_1, (\mu \cdot \gamma_2, 0, 0, 0, \mathbf{i})) \quad , \quad \mathsf{ct}_{1,3} = \mathsf{FHE.Enc}(\mathsf{pk}_1, (\mu \cdot \gamma_3, 0, 0, 0, \mathbf{i})).$$

3. $\mathcal{B}$ then generates the following proofs for each $\ell \in [0, 3]$:

$$\begin{aligned} \pi_{0,\ell} &= \mathsf{NIZK.Prove}(\mathsf{crs}_0, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_{0,\ell}, \mathsf{ct}_{1,\ell}, \mathbf{i}, \{z_{j,1}\}_{j\in[0,n]}), (j, \mathsf{wit}_{z_{j,1}})), \\ \pi_{1,\ell} &= \mathsf{NIZK.Prove}(\mathsf{crs}_1, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_{0,\ell}, \mathsf{ct}_{1,\ell}, \mathbf{i}, y), \mathsf{wit}_y). \end{aligned}$$

4. Finally, for each $\ell \in [0, 3]$, $\mathcal{B}$ generates the encoding $[\alpha_\ell]$ as:

$$[\alpha_\ell] = \Big(\mathsf{ct}_{0,\ell}, \mathsf{ct}_{1,\ell}, \mathbf{i}, \pi_{0,\ell}, \pi_{1,\ell}\Big).$$

$\mathcal{B}$ then provides $\mathcal{A}$ with the MMap public parameters and the challenge encodings. Eventually, $\mathcal{A}$ outputs a bit $b^\star$. $\mathcal{B}$ outputs the same bit $b^\star$. Now, observe the following:

- Suppose that $\mathcal{B}$ received from the challenger the FHE encryptions of the first plaintext tuple in each pair under pk_0. In this case, the view of $\mathcal{A}$ is exactly as in inner hybrid 1-1.

- On the other hand, suppsose that $\mathcal{B}$ from the challenger FHE encryptions of the second plaintext tuple in each pair under pk_0. In this case, the view of $\mathcal{A}$ is exactly as in inner hybrid 1-2.

Hence the advantage of $\mathcal{B}$ in breaking the multi-challenge CPA-security of the FHE scheme is the same as the advantage of $\mathcal{A}$ in distinguishing the inner hybrids 1-1 and 1-2. This allows us to justify this hop.

Inner Hybrid 1-3. In this hybrid, we make the following alterations:

1. We switch the strings crs_0 and crs_1 back to binding mode from hiding mode.

2. We switch the element y in the public parameters back to a non-member from a member for $\mathcal{L}$, i.e., we now sample $y \leftarrow \mathcal{X} \notin \mathcal{L}$.

3. We switch back the circuits for addition, inversion, multiplication and extraction as follows:

$$\begin{aligned} \widetilde{\mathsf{C}}_{\mathsf{Add},0} &\mapsto \widehat{\mathsf{C}}_{\mathsf{Add}} \quad , \quad \widetilde{\mathsf{C}}_{\mathsf{Inv},0} \mapsto \widehat{\mathsf{C}}_{\mathsf{Inv}}, \\ \widetilde{\mathsf{C}}_{\mathsf{Mult},0} &\mapsto \widehat{\mathsf{C}}_{\mathsf{Mult}} \quad , \quad \widetilde{\mathsf{C}}_{\mathsf{ext},0} \mapsto \widehat{\mathsf{C}}_{\mathsf{ext}}. \end{aligned}$$

4. Additionally, we make the following change to the manner in which the challenge encodings are generated: for each encoding, the NIZK proof π_1 under crs_1 now proves that the encoding is consistent using the witness $(\mathsf{sk}_0, \mathsf{sk}_1)$.

Lemma 6.5. *The inner hybrids 1-2 and 1-3 are computationally indistinguishable provided that all of the following assumptions hold:*

- *The piO scheme is indistinguishability-secure against X-IND samplers.*

- *Membership in $\mathcal{L}$ is computationally hard to decide,*

- *Members of $\mathcal{L}$ have unique witnesses for membership.*

Inner Hybrid	$(\mathsf{crs}_0, \mathsf{crs}_1)$	MMap Circuits	y	Challenge Encodings				Remark
				SXDH/Random	Representation	Witness for π_0	Witness for π_1	
2-0	(Binding, Binding)	$\widehat{\mathsf{C}}_{\mathsf{op}}$	$y \notin \mathcal{L}$	Random	Partially oblique	$(j, \mathsf{wit}_{z_{j,1}})$	$(\mathsf{sk}_0, \mathsf{sk}_1)$	
2-1	(Binding, Hiding)	$\widetilde{\mathsf{C}}_{\mathsf{op},1}$	$y \in \mathcal{L}$	Random	Partially oblique	$(j, \mathsf{wit}_{z_{j,1}})$	wit_y	Analogue of Lemma 6.4 for $\beta = 1$
2-2	(Binding, Hiding)	$\widetilde{\mathsf{C}}_{\mathsf{op},1}$	$y \in \mathcal{L}$	Random	Oblique	$(j, \mathsf{wit}_{z_{j,1}})$	wit_y	FHE CPA-security
2-3	(Binding, Binding)	$\widehat{\mathsf{C}}_{\mathsf{op}}$	$y \notin \mathcal{L}$	Random	Oblique	$(j, \mathsf{wit}_{z_{j,1}})$	$(\mathsf{sk}_0, \mathsf{sk}_1)$	Analogue of Lemma 6.5 for $\beta = 1$

Table 12: Overview of the inner hybrids in the proof of indistinguishability of outer hybrids 2 and 3. Changes between subsequent hybrids are highlighted in red. Throughout, $z_{j,1} \in \mathcal{L}$, and (g_0, g_1, g_2, g_3) is a random tuple of group elements. We use the shorthands $\widehat{\mathsf{C}}_{\mathsf{op}}$ and $\widetilde{\mathsf{C}}_{\mathsf{op},1}$ for the tuples $(\widehat{\mathsf{C}}_{\mathsf{Add}}, \widehat{\mathsf{C}}_{\mathsf{Inv}}, \widehat{\mathsf{C}}_{\mathsf{Mult}}, \widehat{\mathsf{C}}_{\mathsf{ext}})$ and $(\widetilde{\mathsf{C}}_{\mathsf{Add}}, \widetilde{\mathsf{C}}_{\mathsf{Inv}}, \widetilde{\mathsf{C}}_{\mathsf{Mult}}, \widetilde{\mathsf{C}}_{\mathsf{ext},1})$, respectively.

- *The dual-mode NIZK proof system satisfies CRS indistinguishability, perfect extractability in the binding mode, and perfect witness-indistinguishability in the hiding mode.*

Proof. The proof of this lemma uses essentially the same inner hybrids as the proof of Lemma 6.4, albeit in the reverse order, and is hence not detailed. □

Observe that in inner hybrid 1-3, the public parameters of the MMap and the challenge encodings are distributed exactly as in outer hybrid 2. This concludes the proof of Lemma 6.3.

Outer Hybrids 2 and 3. We now argue that outer hybrids 2 and 3 are computationally indistinguishable from each other. In particular, we state and prove the following lemma:

Lemma 6.6. *The outer hybrids 2 and 3 are computationally indistinguishable provided that all of the following assumptions hold:*

- *The piO scheme is indistinguishability-secure against X-IND samplers.*
- *Membership in $\mathcal{L}$ is computationally hard to decide,*
- *Members of $\mathcal{L}$ have unique witnesses for membership.*
- *The dual-mode NIZK proof system satisfies CRS indistinguishability, perfect extractability in the binding mode, and perfect witness-indistinguishability in the hiding mode.*

Proof. To prove this lemma, we use another sequence of inner hybrids. The first of these inner hybrids is identical to outer hybrid 2, while the last is identical to outer hybrid 3. Table 12 provides an overview of these inner hybrids. As is evident from the description of the hybrids, they are essentially identical to the inner hybrids used in the proof of Lemma 6.3, with the exception that we need to use the modified circuits $\widetilde{\mathsf{C}}_{\mathsf{Add},1}$, $\widetilde{\mathsf{C}}_{\mathsf{Inv},1}$, $\widetilde{\mathsf{C}}_{\mathsf{Mult},1}$, and $\widetilde{\mathsf{C}}_{\mathsf{ext},1}$ (hardwired with only sk_0 and not sk_1), and we rely on analogous versions of Lemma 6.4 and Lemma 6.5 for $\beta = 1$ as opposed to $\beta = 0$. Apart from this, the proof of indistinguishability of these inner hybrids are essentially identical to those in the proof of Lemma 6.3, and are hence not detailed. □

Inner Hybrid	crs_0	MMap Circuits	$z_{j,0}$	$z_{j,1}$	Challenge Encodings				Remark
					SXDH/Random	Representation	Witness for π_0	Witness for π_1	
3-0	Binding	$\widehat{\mathsf{C}}_{\mathsf{op}}$	$z_{j,0} \in \mathcal{L}$	$z_{j,1} \in \mathcal{L}$	Random	Normal	$(j, \mathsf{wit}_{z_{j,1}})$	$(\mathsf{sk}_0, \mathsf{sk}_1)$	NIZK CRS-indistinguishability
3-1	Binding	C_{op}	$z_{j,0} \in \mathcal{L}$	$z_{j,1} \in \mathcal{L}$	Random	Normal	$(j, \mathsf{wit}_{z_{j,1}})$	$(\mathsf{sk}_0, \mathsf{sk}_1)$	piO-security + unique witnesses
3-2	Binding	C_{op}	$z_{j,0} \notin \mathcal{L}$	$z_{j,1} \in \mathcal{L}$	Random	Normal	$(j, \mathsf{wit}_{z_{j,1}})$	$(\mathsf{sk}_0, \mathsf{sk}_1)$	$\mathcal{L}$-hardness
3-3	Binding	C_{op} except Hardwired witness for $z_{j,1}$	$z_{j,0} \notin \mathcal{L}$	$z_{j,1} \in \mathcal{L}$	Random	Normal	$(j, \mathsf{wit}_{z_{j,1}})$	$(\mathsf{sk}_0, \mathsf{sk}_1)$	$\mathcal{L}$-hardness
3-4	Binding	$\widehat{\widehat{\mathsf{C}}}_{\mathsf{op}}$	$z_{j,0} \notin \mathcal{L}$	$z_{j,1} \in \mathcal{L}$	Random	Normal	$(j, \mathsf{wit}_{z_{j,1}})$	$(\mathsf{sk}_0, \mathsf{sk}_1)$	piO-security + NIZK perfect soundness

Table 13: Overview of the inner hybrids in the proof of indistinguishability of outer hybrids 3 and 4. Changes between subsequent hybrids are highlighted in red. Throughout, crs_1 is in binding mode, $y \notin \mathcal{L}$, and (g_0, g_1, g_2, g_3) is a random tuple of group elements. We use the shorthands $\widehat{\mathsf{C}}_{\mathsf{op}}$ and $\widehat{\widehat{\mathsf{C}}}_{\mathsf{op}}$ for the tuples $(\widehat{\mathsf{C}}_{\mathsf{Add}}, \widehat{\mathsf{C}}_{\mathsf{Inv}}, \widehat{\mathsf{C}}_{\mathsf{Mult}}, \widehat{\mathsf{C}}_{\mathsf{ext}})$ and $(\widehat{\mathsf{C}}_{\mathsf{Add}}, \widehat{\mathsf{C}}_{\mathsf{Inv}}, \widehat{\widehat{\mathsf{C}}}_{\mathsf{op}}, \widehat{\mathsf{C}}_{\mathsf{ext}})$, respectively, where the second set of circuits are described in detail subsequently.

Outer Hybrids 3 and 4. We now argue that outer hybrids 3 and 4 are computationally indistinguishable from each other. In particular, we state and prove the following lemma:

Lemma 6.7. *The outer hybrids 3 and 4 are computationally indistinguishable provided that all of the following assumptions hold:*

- *The piO scheme is indistinguishability-secure against X-IND samplers.*
- *Membership in $\mathcal{L}$ is computationally hard to decide,*
- *Members of $\mathcal{L}$ have unique witnesses for membership.*
- *The dual-mode NIZK proof system satisfies CRS indistinguishability, perfect extractability in the binding mode, and perfect witness-indistinguishability in the hiding mode.*

Proof. To prove this lemma, we use another sequence of inner hybrids. The first of these inner hybrids is identical to outer hybrid 3, while the last is identical to outer hybrid 4. Table 13 provides an overview of these inner hybrids.

Inner Hybrid 3-0. This hybrid is identical to inner hybrid 3.

Inner Hybrid 3-1. In this hybrid we switch back to using the original MMap circuits $\mathsf{C}_{\mathsf{Add}}$, $\mathsf{C}_{\mathsf{Inv}}$, $\mathsf{C}_{\mathsf{Mult}}$ and $\mathsf{C}_{\mathsf{ext}}$. In particular, we no longer hardwire the witnesses $\mathsf{wit}_{z_{j,0}}$ and $\mathsf{wit}_{z_{j,1}}$ for the membership of $z_{j,0}$ and $z_{j,1}$ into the circuits, and hardwire the extraction trapdoor $\mathsf{t}_{\mathsf{ext},0}$ into all of the circuits. We claim that this change does not change the functionality of the MMap circuits at all.

To begin with, observe that in order to reach the extraction step, the input encoding(s) to each circuit must pass the validity checks for π_0. Since crs_0 is binding in both inner hybrids 3-0 and 3-1, any valid encoding that passes this test must either contain a verifying proof of normal representation, or must contain a verifying proof of language-membership for $z_{j,0}$ or $z_{j,1}$. Now observe the following:

- In the former case, the first **If** condition in the addition and inversion circuits, the **Else If** conditions in the multiplication circuit, and the first **If** condition in the extraction circuit are never satisfied, and the proof π_0^* for the output encoding is generated using the witness $(\mathsf{sk}_0, \mathsf{sk}_1)$. Hence, in this case, the outputs of the original and modified MMap circuits are identical.

- In the latter case, these conditions may be satisfied (if one or more of the input encodings to each circuit are in the oblique representation). In this case, to generate the proof π_0^* for the output encoding, the circuits $\mathsf{C}_{\mathsf{Add}}$, $\mathsf{C}_{\mathsf{Inv}}$ and $\mathsf{C}_{\mathsf{Mult}}$ use the extracted witness, while the circuits $\widehat{\mathsf{C}}_{\mathsf{Add}}$, $\widehat{\mathsf{C}}_{\mathsf{Inv}}$ and $\widehat{\mathsf{C}}_{\mathsf{Mult}}$ use the hardwired witness. Since the NIZK system has perfect extractability in the binding mode, extraction always succeeds in the original MMap circuits. Since $\mathcal{L}$ has unique membership witnesses, the extracted and hardwired witnesses must be identical. Hence, even in this case, the outputs of the original and modified MMap circuits are identical.

At this point, the transition can be justified by invoking the security of the piO scheme against X-IND samplers (once for each MMap circuit).

Inner Hybrid 3-2. In this hybrid, we switch $z_{j,0}$ in the public parameter of the MMap from a uniform member to a uniform non-member of $\mathcal{L}$. By the hardness of deciding membership in the set $\mathcal{L}$, inner hybrid 3-2 is computationally indistinguishable from inner hybrid 3-1.

Inner Hybrid 3-3. In this hybrid, we switch to using a set of circuits that are essentially identical to the MMap circuits $\widehat{\mathsf{C}}_{\mathsf{Add}}$, $\widehat{\mathsf{C}}_{\mathsf{Inv}}$, $\widehat{\mathsf{C}}_{\mathsf{Mult}}$ and $\widehat{\mathsf{C}}_{\mathsf{ext}}$, except that they only hardwire the witness for $z_{j,1}$ (and not both $z_{j,0}$ and $z_{j,1}$). These circuits do not use the extraction trapdoor $\mathsf{t}_{\mathsf{ext},0}$. We claim that this change does not change the functionality of the MMap circuits at all.

To begin with, observe that in order to reach the extraction step, the input encoding(s) to each circuit must pass the validity checks for π_0. Since crs_0 is binding in both inner hybrids 3-2 and 3-3, any valid encoding that passes this test must either contain a verifying proof of normal representation, or must contain a verifying proof of language-membership for $z_{j,1}$ ($z_{j,0}$ is no longer a member of the language $\mathcal{L}$). Now observe the following:

- In the former case, the first **If** condition in the addition and inversion circuits, the **Else If** conditions in the multiplication circuit, and the first **If** condition in the extraction circuit are never satisfied, and the proof π_0^* for the output encoding is generated using the witness $(\mathsf{sk}_0, \mathsf{sk}_1)$. Hence, in this case, the outputs of the original and modified MMap circuits are identical.

- In the latter case, these conditions may be satisfied (if one or more of the input encodings to each circuit are in the oblique representation). In this case, to generate the proof π_0^* for the output encoding, the circuits $\mathsf{C}_{\mathsf{Add}}$, $\mathsf{C}_{\mathsf{Inv}}$ and $\mathsf{C}_{\mathsf{Mult}}$ use the extracted witness, while the circuits $\widehat{\mathsf{C}}_{\mathsf{Add}}$, $\widehat{\mathsf{C}}_{\mathsf{Inv}}$ and $\widehat{\mathsf{C}}_{\mathsf{Mult}}$ use the hardwired witness. Since the NIZK system has perfect extractability in the binding mode, extraction always succeeds in the original MMap circuits. Since $\mathcal{L}$ has unique membership witnesses, the extracted and hardwired witnesses must be identical. Hence, even in this case, the outputs of the original and modified MMap circuits are identical. □

At this point, the transition can be justified by invoking the security of the piO scheme against X-IND samplers (once for each MMap circuit). Note that by this hybrid, the addition and inversion circuits are already identical to $\widehat{\widehat{\mathsf{C}}}_{\mathsf{Add}}$ and $\widehat{\widehat{\mathsf{C}}}_{\mathsf{Inv}}$, respectively, while the extraction circuit is identical to $\widehat{\mathsf{C}}_{\mathsf{ext}}$, exactly as in inner hybrid 3-4.

Inner Hybrid 3-4. In this hybrid, we switch the multiplication circuit to $\widehat{\widehat{\mathsf{C}}}_{\mathsf{Mult}}$ (Figure 14). In particular, we switch to using the modified homomorphic evaluation circuit $\widehat{\widehat{\mathsf{C}}}_{\mathsf{Mult,FHE}}$ (Figure 13). We also remove the hardwired secret exponents, i.e., the multiplication circuit is now hardwired with (g_0, g_1, g_2, g_3) instead of $(g_0, \gamma_1, \gamma_2, \gamma_3)$. We claim that this does not change the functionality of the multiplication circuit at all.

We claim that the output of the homomorphic evaluation circuit does not change despite the modifications made to it. Observe that in order to reach the **Else If** statement in Step 3 of the homomorphic evaluation circuit $\widehat{\widehat{\mathsf{C}}}_{\mathsf{Mult}}$ (Figure 13), at least one input encoding needs to be in oblique representation with a verifying proof of language membership for $zj, 0$. However, the input encoding(s) must also pass the validity checks for π_0 prior to the homomorphic evaluation. Since crs_0 is binding in both inner hybrids 3-3 and 3-4, any valid encoding that passes this test must either contain a verifying proof of normal representation, or must contain a verifying proof of language-membership for $z_{j,1}$ ($z_{j,0}$ is no longer a member of the language $\mathcal{L}$). This immediately implies that Step 3 of the homomorphic evaluation circuit cannot be reached by any pair of valid input encodings. Hence, changing this step by removing the secret exponents from it does not change the output of the homomorphic circuit.

At this point, we can argue that the output of the overall multiplication circuit $\widehat{\widehat{\mathsf{C}}}_{\mathsf{Mult}}$ also remains unchanged (since the output proofs are still generated in exactly the same way), and the transition can be justified by invoking the security of the piO scheme against X-IND samplers.

Observe that in inner hybrid 3-4, the public parameters of the MMap and the challenge encodings are distributed exactly as in outer hybrid 4. This concludes the proof of Lemma 6.7.

Outer Hybrids 4 and 5. We state and prove the following lemma:

Lemma 6.8. *The outer hybrids 4 and 5 are computationally indistinguishable provided that the DDH assumption holds over the group* $\mathbb{G}$.

Proof. Suppose that there exists a PPT adversary $\mathcal{A}$ that can distinguish between the outer hybrids 4 and 5 with non-negligible probability. We construct a PPT algorithm $\mathcal{B}$ that can break the DDH assumption over the group $\mathbb{G}$ with non-negligible probability. As part of its input DDH challenge, $\mathcal{B}$ receives a tuple of group elements of the form (g_0, g_1, g_2, g_3), and sets up the public parameters for the MMap as follows:

1. $\mathcal{B}$ samples two key-pairs for the FHE scheme as:
$$(\mathsf{pk}_0, \mathsf{sk}_0), (\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathsf{FHE.Gen}(1^\lambda),$$
and a pair of binding NIZK CRS strings (along with the corresponding extraction trapdoors) as:
$$(\mathsf{crs}_0, \mathsf{t}_{\mathsf{ext},0}), (\mathsf{crs}_1, \mathsf{t}_{\mathsf{ext},1}) \leftarrow \mathsf{NIZK.Setup}(1^\lambda, \mathsf{binding}).$$

2. Next $\mathcal{B}$ uniformly samples a total $(n+1)$ elements from the set $\mathcal{X}$ that are all non-members for the subset $\mathcal{L}$ and one element that is a member for $\mathcal{L}$. More formally, it samples
$$y, z_{1,0}, z_{1,1}, \ldots, z_{j-1,0}, z_{j-1,1}, z_{j,0}, z_{j+1,0}, z_{j+1,1} \ldots, z_{n,0}, z_{n,1} \leftarrow \mathcal{X} \setminus \mathcal{L},$$
$$z_{j,1} \leftarrow \mathcal{L},$$
where $z_{j,1}$ has unique membership-witness $\mathsf{wit}_{z_{j,1}}$.

3. Finally, $\mathcal{B}$ sets up the MMap circuits $\widehat{\widehat{\mathsf{C}}}_{\mathsf{Add}}$, $\widehat{\widehat{\mathsf{C}}}_{\mathsf{Inv}}$, $\widehat{\widehat{\mathsf{C}}}_{\mathsf{Mult}}$ and $\widehat{\mathsf{C}}_{\mathsf{ext}}$ exactly as described in Figures 11, 12, 14 and 10, respectively, except for the fact that it hardwires its input tuple (g_0, g_1, g_2, g_3) into each of these circuits. It then computes and outputs four probabilistically indistinguishability-obfuscated circuits of the form
$$\bar{\mathsf{C}}_{\mathsf{Add}} = \mathsf{piO.Obf}(\widehat{\widehat{\mathsf{C}}}_{\mathsf{Add}}) \quad , \quad \bar{\mathsf{C}}_{\mathsf{Inv}} = \mathsf{piO.Obf}(\widehat{\widehat{\mathsf{C}}}_{\mathsf{Inv}}),$$
$$\bar{\mathsf{C}}_{\mathsf{Mult}} = \mathsf{piO.Obf}(\widehat{\widehat{\mathsf{C}}}_{\mathsf{Mult}}) \quad , \quad \bar{\mathsf{C}}_{\mathsf{ext}} = \mathsf{piO.Obf}(\widehat{\widehat{\mathsf{C}}}_{\mathsf{ext}}).$$
In particular, note that creating the multiplication circuit $\widehat{\widehat{\mathsf{C}}}_{\mathsf{Mult}}$ does not require hardwiring the actual secret exponents in $\mathbb{Z}_q$; hence the knowledge of the tuple of group elements (g_0, g_1, g_2, g_3) suffices in this case.

Next, $\mathcal{B}$ sets up the challenge encodings in the oblique representation as follows (refer to Table 5 for how the challenge encodings are formatted in oblique representation in outer hybrid 3):

1. $\mathcal{B}$ samples $\mu \leftarrow \mathbb{Z}_q$ and generates the following FHE ciphertexts for each $b \in \{0, 1\}$:
$$\mathsf{ct}_{b,0} = \mathsf{FHE.Enc}(\mathsf{pk}_b, (\mu, 0, 0, 0, \mathbf{i})) \quad , \quad \mathsf{ct}_{b,1} = \mathsf{FHE.Enc}(\mathsf{pk}_b, (0, \mu, 0, 0, \mathbf{i})),$$
$$\mathsf{ct}_{b,2} = \mathsf{FHE.Enc}(\mathsf{pk}_b, (0, 0, \mu, 0, \mathbf{i})) \quad , \quad \mathsf{ct}_{b,3} = \mathsf{FHE.Enc}(\mathsf{pk}_b, (0, 0, 0, \mu, \mathbf{i})).$$

2. $\mathcal{B}$ generates the following proofs for each $\ell \in [0,3]$:

$$\begin{aligned}\pi_{0,\ell} &= \mathsf{NIZK.Prove}(\mathsf{crs}_0, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_{0,\ell}, \mathsf{ct}_{1,\ell}, \mathbf{i}, \{z_{j,1}\}_{j\in[0,n]}), (j, \mathsf{wit}_{z_{j,1}})),\\ \pi_{1,\ell} &= \mathsf{NIZK.Prove}(\mathsf{crs}_1, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_{0,\ell}, \mathsf{ct}_{1,\ell}, \mathbf{i}, y), (\mathsf{sk}_0, \mathsf{sk}_1)).\end{aligned}$$

3. Finally, for each $\ell \in [0,3]$, $\mathcal{B}$ generates the encoding $[\alpha_\ell]$ as:

$$[\alpha_\ell] = \Big(\mathsf{ct}_{0,\ell}, \mathsf{ct}_{1,\ell}, \mathbf{i}, \pi_{0,\ell}, \pi_{1,\ell}\Big).$$

$\mathcal{B}$ then provides $\mathcal{A}$ with the MMap public parameters and the challenge encodings. Eventually, $\mathcal{A}$ outputs a bit $b^\star$. $\mathcal{B}$ outputs the same bit $b^\star$. Now, observe the following:

- Suppose that $\mathcal{B}$ receives as input a tuple of uniformly random group elements of the form $(g_0, g_0^{\gamma_1}, g_0^{\gamma_2}, g_0^{\gamma_3})$. In this case, the view of $\mathcal{A}$ is exactly as in outer hybrid 3.
- On the other hand, when $\mathcal{B}$ receives as input a DDH tuple of the form $(g_0, g_0^{\gamma_1}, g_0^{\gamma_2}, g_0^{\gamma_1\cdot\gamma_2})$, the view of $\mathcal{A}$ is exactly as in outer hybrid 4. □

Hence the advantage of $\mathcal{B}$ in breaking DDH over the group $\mathbb{G}$ is the same as the advantage of $\mathcal{A}$ in distinguishing the outer hybrids 3 and 4. This concludes the proof of Lemma 6.7.

Outer Hybrids 5 and 6. We now argue that outer hybrids 5 and 6 are computationally indistinguishable from each other. In particular, we state and prove the following lemma:

Lemma 6.9. *The outer hybrids 5 and 6 are computationally indistinguishable provided that all of the following assumptions hold:*

- *The piO scheme is indistinguishability-secure against X-IND samplers.*
- *Membership in $\mathcal{L}$ is computationally hard to decide,*
- *Members of $\mathcal{L}$ have unique witnesses for membership.*
- *The dual-mode NIZK proof system satisfies CRS indistinguishability, perfect extractability in the binding mode, and perfect witness-indistinguishability in the hiding mode.*

Proof. The proof of this lemma uses essentially the same inner hybrids as in the proof of Lemma 6.7, albeit in reverse order, and is hence not detailed. □

Outer Hybrids 6 and 7. We state and prove the following lemma:

Lemma 6.10. *The outer hybrids 6 and 7 are computationally indistinguishable provided that all of the following assumptions hold:*

- *The piO scheme is indistinguishability-secure against X-IND samplers.*
- *Membership in $\mathcal{L}$ is computationally hard to decide,*
- *Members of $\mathcal{L}$ have unique witnesses for membership.*
- *The dual-mode NIZK proof system satisfies CRS indistinguishability, perfect extractability in the binding mode, and perfect witness-indistinguishability in the hiding mode.*

Proof. The proof of this lemma uses essentially the same inner hybrids as in the proof of Lemma 6.6, albeit in reverse order, and is hence not detailed. □

Outer Hybrids 7 and 8. We state the following lemma:

Lemma 6.11. *The outer hybrids 7 and 8 are computationally indistinguishable provided that all of the following assumptions hold:*

- *The piO scheme is indistinguishability-secure against X-IND samplers.*
- *Membership in $\mathcal{L}$ is computationally hard to decide,*
- *Members of $\mathcal{L}$ have unique witnesses for membership.*
- *The dual-mode NIZK proof system satisfies CRS indistinguishability, perfect extractability in the binding mode, and perfect witness-indistinguishability in the hiding mode.*

Proof. The proof of this lemma uses essentially the same inner hybrids as in the proof of Lemma 6.3, albeit in reverse order, and is hence not detailed. □

Outer Hybrids 8 and 9. We state the following lemma:

Lemma 6.12. *The outer hybrids 8 and 9 are computationally indistinguishable provided that all of the following assumptions hold:*

- *The piO scheme is indistinguishability-secure against X-IND samplers.*
- *Membership in $\mathcal{L}$ is computationally hard to decide,*
- *Members of $\mathcal{L}$ have unique witnesses for membership.*
- *The dual-mode NIZK proof system satisfies CRS indistinguishability, perfect extractability in the binding mode, and perfect witness-indistinguishability in the hiding mode*

Proof. The proof of this lemma uses essentially the same inner hybrids as in the proof of Lemma 6.2, albeit in reverse order, and is hence not detailed. □

Outer Hybrids 9 and 10. We state and prove the following lemma:

Lemma 6.13. *The outer hybrids 9 and 10 are computationally indistinguishable provided that all of the following assumptions hold:*

- *The piO scheme is indistinguishability-secure against X-IND samplers.*
- *The dual-mode NIZK proof system satisfies perfect soundness in the binding mode.*

Proof. In this hybrid, we switch the tuple of group elements (g_0, g_1, g_2, g_3) hardwired into the MMap circuits as follows: we switch the the tuple of exponents $(\gamma_1, \gamma_2, \gamma_3 = \gamma_1 \cdot \gamma_2)$ from a triplet corresponding to a DDH tuple to a triplet corresponding to a random tuple, i.e., γ_3 is now sampled uniformly at random from $\mathbb{Z}_q$. Effectively, this switches the tuple of group elements (g_0, g_1, g_2, g_3) from a uniform DDH tuple to a uniformly random tuple, albeit with the same base element g_0.

We argue below that this switch does not alter the output distribution of these circuits from outer hybrid 9 to outer hybrid 10. Once this is established, the indistinguishability argument follows immediately under the assumption that the piO scheme is indistinguishability-secure against X-IND samplers (once for each MMap circuit).

We first focus on the MMap circuits $\mathsf{C}_{\mathsf{Add}}$ and $\mathsf{C}_{\mathsf{Inv}}$. Observe that switching (g_0, g_1, g_2, g_3) from a uniform DDH tuple to a uniformly random tuple only (potentially) affects the outcome of the "**If**" condition in step 7 of each of these circuits. Note however that in both outer hybrids 9 and 10, crs_0 is in binding mode, each $z_{j,b}$ in the public parameter is a non-member for $j \in [n]$ and $b \in \{0, 1\}$, and y in the public parameter is also a non-member. Hence, it follows from the perfect soundness of the dual-mode NIZK proof system that any input encoding(s) for which these circuits do not output $\perp$ and terminate before step 7 must be *both* in normal representation *and* consistent. This in turn guarantees that the outcome of the "**If**" condition in step 7 must be "false". Hence, the output distributions of the MMap circuits $\mathsf{C}_{\mathsf{Add}}$ and $\mathsf{C}_{\mathsf{Inv}}$ in outer hybrids 9 and 10 are identical.

Next, we focus on the MMap circuit $\mathsf{C}_{\mathsf{Mult}}$. Observe that switching (g_0, g_1, g_2, g_3) from a uniform DDH tuple to a uniformly random tuple only (potentially) affects the outcome of the "**Else If**" condition in step 3 of the circuit $\mathsf{C}_{\mathsf{Mult,FHE}}$ (Figure 4), which is evaluated homomorphically in Step 5 of $\mathsf{C}_{\mathsf{Mult}}$. Note however that in both outer hybrids 9 and 10, crs_0 is in binding mode, each $z_{j,b}$ in the public parameter is a non-member for $j \in [n]$ and $b \in \{0, 1\}$, and y in the public parameter is also a non-member. Hence, it follows from the perfect soundness of the dual-mode NIZK proof system that any input encoding(s) for which these circuits do not output $\perp$ and terminate before step 7 must be *both* in normal representation *and* consistent. Hence, the "**Else If**" condition in step 3 of the circuit $\mathsf{C}_{\mathsf{Mult,FHE}}$ can never be satisfied by a pair of valid input encodings. This in turn implies that the output distributions of the MMap circuit $\mathsf{C}_{\mathsf{Mult}}$ in outer hybrids 9 and 10 are identical.

Finally, we focus on the MMap extraction circuit $\mathsf{C}_{\mathsf{ext}}$. Observe that switching (g_0, g_1, g_2, g_3) from a uniform DDH tuple to a uniformly random tuple potentially affects the output of the extraction circuit. However, note yet again that in both outer hybrids 9 and 10, crs_0 is in binding mode, each $z_{j,b}$ in the public parameter is a non-member for $j \in [n]$ and $b \in \{0, 1\}$, and y in the public parameter is also a non-member. Hence, it follows from the perfect soundness of the dual-mode NIZK proof system that any input encoding(s) for which the circuit does not output $\perp$ and terminate before the extraction step is executed must be *both* in normal representation *and* consistent. Observe also that the base element g_0 in the tuple of group elements remains unaltered across outer hybrids 9 and 10. It follows immediately that the outcome of extraction on a given encoding in normal representation in both hybrids is identical. In other words, the output distributions of $\mathsf{C}_{\mathsf{ext}}$ in outer hybrids 9 and 10 are identical.

This concludes the proof of Lemma 6.13, and hence the proof of Theorem 6.1. □

7 Achieving Exponent-DDH Hardness

In this section, we prove that for any $k \geq 2$, solving k-EDDH is hard over our proposed MMap construction if solving k-EDDH is hard over the group $\mathbb{G}$. More specifically we state and prove the following theorem:

Theorem 7.1. *For any $k \geq 2$, the k-EDDH assumption holds over our proposed MMap construction provided that all of the following assumptions hold:*

- *The piO scheme is indistinguishability-secure against X-IND samplers.*
- *Membership in $\mathcal{L}$ is computationally hard to decide,*
- *Members of $\mathcal{L}$ have unique witnesses for membership.*
- *The dual-mode NIZK proof system satisfies CRS indistinguishability, perfect extractability in the binding mode, and perfect witness-indistinguishability in the hiding mode.*
- *The k-EDDH assumption holds over the group $\mathbb{G}$.*

The proof shares a very similar hybrid structure with the proof of SXDH-hardness in Appendix 6. We outline the outer hybrids here for the sake of completeness. The workings of the inner hybrids are almost identical to that in the proof of SXDH, and are hence not detailed.

7.1 Outer Hybrids

We begin by describing the outer hybrids for our proof. Outer hybrid 0 corresponds to the game where the challenger provides the adversary with encodings of uniformly random elements in a level-set that is chosen by the adversary. The final outer hybrid corresponds to the game where the challenger provides the adversary with a valid k-EDDH instance, i.e., encodings of elements sampled according to the real k-EDDH distribution, in the same level-set chosen by the adversary. Before we describe the outer hybrids, we describe a few conditions that remain invariant throughout the outer hybrids:

Outer Hybrid	MMap Circuits	$z_{j,0}$	$z_{j,1}$	(g_0, g_1, g_2)	Challenge Encodings			
					k-EDDH/Random	Representation	Witness for π_0	Witness for π_1
0	C_{op}	$z_{j,0} \notin \mathcal{L}$	$z_{j,1} \notin \mathcal{L}$	Random	Random	Normal	$(\mathsf{sk}_0, \mathsf{sk}_1)$	$(\mathsf{sk}_0, \mathsf{sk}_1)$
1	$\widehat{\mathsf{C}}_{\mathsf{op}}$	$z_{j,0} \in \mathcal{L}$	$z_{j,1} \in \mathcal{L}$	Random	Random	Normal	$(j, \mathsf{wit}_{z_{j,1}})$	$(\mathsf{sk}_0, \mathsf{sk}_1)$
2	$\widehat{\mathsf{C}}_{\mathsf{op}}$	$z_{j,0} \in \mathcal{L}$	$z_{j,1} \in \mathcal{L}$	Random	Random	Partially oblique	$(j, \mathsf{wit}_{z_{j,1}})$	$(\mathsf{sk}_0, \mathsf{sk}_1)$
3	$\widehat{\mathsf{C}}_{\mathsf{op}}$	$z_{j,0} \in \mathcal{L}$	$z_{j,1} \in \mathcal{L}$	Random	Random	Oblique	$(j, \mathsf{wit}_{z_{j,1}})$	$(\mathsf{sk}_0, \mathsf{sk}_1)$
4	$\widehat{\widehat{\mathsf{C}}}_{\mathsf{op}}$	$z_{j,0} \notin \mathcal{L}$	$z_{j,1} \in \mathcal{L}$	Random	Random	Oblique	$(j, \mathsf{wit}_{z_{j,1}})$	$(\mathsf{sk}_0, \mathsf{sk}_1)$
5	$\widehat{\widehat{\mathsf{C}}}_{\mathsf{op}}$	$z_{j,0} \notin \mathcal{L}$	$z_{j,1} \in \mathcal{L}$	k-EDDH	k-EDDH	Oblique	$(j, \mathsf{wit}_{z_{j,1}})$	$(\mathsf{sk}_0, \mathsf{sk}_1)$
6	$\widehat{\mathsf{C}}_{\mathsf{op}}$	$z_{j,0} \in \mathcal{L}$	$z_{j,1} \in \mathcal{L}$	k-EDDH	k-EDDH	Oblique	$(j, \mathsf{wit}_{z_{j,1}})$	$(\mathsf{sk}_0, \mathsf{sk}_1)$
7	$\widehat{\mathsf{C}}_{\mathsf{op}}$	$z_{j,0} \in \mathcal{L}$	$z_{j,1} \in \mathcal{L}$	k-EDDH	k-EDDH	Partially oblique	$(j, \mathsf{wit}_{z_{j,1}})$	$(\mathsf{sk}_0, \mathsf{sk}_1)$
8	$\widehat{\mathsf{C}}_{\mathsf{op}}$	$z_{j,0} \in \mathcal{L}$	$z_{j,1} \in \mathcal{L}$	k-EDDH	k-EDDH	Normal	$(j, \mathsf{wit}_{z_{j,1}})$	$(\mathsf{sk}_0, \mathsf{sk}_1)$
9	C_{op}	$z_{j,0} \notin \mathcal{L}$	$z_{j,1} \notin \mathcal{L}$	k-EDDH	k-EDDH	Normal	$(\mathsf{sk}_0, \mathsf{sk}_1)$	$(\mathsf{sk}_0, \mathsf{sk}_1)$
10	C_{op}	$z_{j,0} \notin \mathcal{L}$	$z_{j,1} \notin \mathcal{L}$	Random	k-EDDH	Normal	$(\mathsf{sk}_0, \mathsf{sk}_1)$	$(\mathsf{sk}_0, \mathsf{sk}_1)$

Table 14: Overview of the outer hybrids in the proof of k-EDDH. Changes between subsequent hybrids are highlighted in red. Throughout, crs_0 and crs_1 are binding, $y \notin \mathcal{L}$, the element g_3 in the tuple (g_0, g_1, g_2, g_3) is sampled uniformly from the group $\mathbb{G}$, and the challenge encodings are consistent with respect to extraction. We use the shorthands C_{op} and $\widehat{\mathsf{C}}_{\mathsf{op}}$ for the tuples $(\mathsf{C}_{\mathsf{Add}}, \mathsf{C}_{\mathsf{Inv}}, \mathsf{C}_{\mathsf{Mult}}, \mathsf{C}_{\mathsf{ext}})$ and $(\widehat{\mathsf{C}}_{\mathsf{Add}}, \widehat{\mathsf{C}}_{\mathsf{Inv}}, \widehat{\mathsf{C}}_{\mathsf{Mult}}, \widehat{\mathsf{C}}_{\mathsf{ext}})$, respectively, where the second set of circuits are described in detail subsequently.

- The NIZK CRS strings crs_0 and crs_1 are in binding mode, as in the real MMap scheme, in all the outer hybrids.
- The element y in the public parameter is a non-member for the language $\mathcal{L}$, as in the real MMap scheme, in all the outer hybrids.
- The challenge encodings provided to the adversary are *consistent* with respect to extraction (as formalized by the relation R_1 described earlier) in all the outer hybrids. However, as we shall see later, they may be switched from the normal to oblique representation and vice-versa.
- The term g_3 in the tuple (g_0, g_1, g_2, g_3) hardwired into the MMap circuits is sampled uniformly throughout.

Table 1 provides an overview of the outer hybrids, which we now describe in details.
Table 14 provides an overview of the outer hybrids, which we now describe in details.

Outer Hybrid 0. In this hybrid, the MMap is set up exactly as in the real scheme described earlier. Let (g_0, g_1, g_2, g_3) be the tuple of group elements hardwired into each of the MMap circuits, such that $g_\ell = g_0^{\gamma_\ell}$ for $\ell \in \{1, 2, 3\}$. Let μ be a uniformly sampled element in $\mathbb{Z}_q$. The SXDH adversary is provided with encodings of the tuple of elements:

$$(\alpha_0, \alpha_1, \alpha_2) = (\mu, \mu \cdot \gamma_1, \mu \cdot \gamma_2),$$

Encoding	ct_0 encrypts	ct_1 encrypts	Witness for π_0	Witness for π_1
$[\alpha_0]$	$(\mu, 0, 0, 0, \mathbf{i})$	$(\mu, 0, 0, 0, \mathbf{i})$	$(\mathsf{sk}_0, \mathsf{sk}_1)$	$(\mathsf{sk}_0, \mathsf{sk}_1)$
$[\alpha_1]$	$(\mu \cdot \gamma_1, 0, 0, 0, \mathbf{i})$	$(\mu \cdot \gamma_1, 0, 0, 0, \mathbf{i})$	$(\mathsf{sk}_0, \mathsf{sk}_1)$	$(\mathsf{sk}_0, \mathsf{sk}_1)$
$[\alpha_2]$	$(\mu \cdot \gamma_2, 0, 0, 0, \mathbf{i})$	$(\mu \cdot \gamma_2, 0, 0, 0, \mathbf{i})$	$(\mathsf{sk}_0, \mathsf{sk}_1)$	$(\mathsf{sk}_0, \mathsf{sk}_1)$

Table 15: Overview of Challenge Encodings in Outer Hybrid 0

where the encodings are generated in normal form (formalized by relation R_0 described in Appendix 5) corresponding to the level-set $\mathbf{i}$ chosen by the k-EDDH adversary. For ease of understanding, we explicitly lay out what the FHE ciphertexts in each of the encodings actually encrypt in Table 15. Along with the FHE encryptions, each encoding also contains a NIZK proof π_0 for normal representation under the binding crs_0, and a NIZK proof π_1 for consistency with respect to extraction under the binding crs_1.

Outer Hybrid 1. In this hybrid, we make the following alterations to the manner in which the MMap is generated:

1. Fix $j \in [n]$ such that for the challenge level-set $\mathbf{i}$, we have $\mathbf{i}_j = 1$. We switch the elements $z_{j,0}$ and $z_{j,1}$ in the public parameters from non-members to members for $\mathcal{L}$, i.e., we now sample $z_{j,0}, z_{j,1} \leftarrow \mathcal{L}$, along with their (unique) membership-witnesses $\mathsf{wit}_{z_{j,0}}$ and $\mathsf{wit}_{z_{j,1}}$, respectively.

2. We switch the circuits for addition, inversion, multiplication and extraction as follows:

$$\begin{aligned} \mathsf{C}_{\mathsf{Add}} \mapsto \widehat{\mathsf{C}}_{\mathsf{Add}} \quad &, \quad \mathsf{C}_{\mathsf{Inv}} \mapsto \widehat{\mathsf{C}}_{\mathsf{Inv}}, \\ \mathsf{C}_{\mathsf{Mult}} \mapsto \widehat{\mathsf{C}}_{\mathsf{Mult}} \quad &, \quad \mathsf{C}_{\mathsf{ext}} \mapsto \widehat{\mathsf{C}}_{\mathsf{ext}}, \end{aligned}$$

The switched circuits are described in Figures 7, 8, 9, and 10, respectively. At a high level, we make the following alterations (these are essentially identical to those in outer hybrid 1 of the proof of SXDH-hardness):

- We hardwire the witnesses $\mathsf{wit}_{z_{j,0}}$ and $\mathsf{wit}_{z_{j,1}}$ into the modified addition and inversion circuits $\widehat{\mathsf{C}}_{\mathsf{Add}}$ and $\widehat{\mathsf{C}}_{\mathsf{Inv}}$, and remove the extraction trapdoor $\mathsf{t}_{\mathsf{ext},0}$ from both these circuits. Instead of extracting the membership witnesses from the input encodings, we directly use these hardwired witnesses to generate proofs of membership for the output encodings.

- We hardwire the witness $\mathsf{wit}_{z_{j,1}}$ into the modified multiplication circuit $\widehat{\mathsf{C}}_{\mathsf{Mult}}$, and remove the extraction trapdoor $\mathsf{t}_{\mathsf{ext},0}$ from it. Instead of extracting the membership witness from the input encoding, we directly use the hardwired witness to generate proofs of membership for the output encodings. Note that we only need one witness for the multiplication circuit; the witness $\mathsf{wit}_{z_{j,0}}$ is not used (the reader may observe that the original multiplication circuit would not have extracted $\mathsf{wit}_{z_{j,0}}$ either).

- Finally, we remove the extraction trapdoor $\mathsf{t}_{\mathsf{ext},0}$ from the modified extraction circuit $\widehat{\mathsf{C}}_{\mathsf{ext}}$

Additionally, we make the following change to the manner in which the challenge encodings are generated: for each challenge encoding, the NIZK proof π_0 now proves (under the binding crs_0) that $z_{j,1} \in \mathcal{L}$ as opposed to proving that the encoding is in the normal representation. This is explicitly described in Table 16.

Encoding	ct_0 **encrypts**	ct_1 **encrypts**	**Witness for** π_0	**Witness for** π_1
$[\alpha_0]$	$(\mu, 0, 0, 0, \mathbf{i})$	$(\mu, 0, 0, 0, \mathbf{i})$	$(j, \mathsf{wit}_{z_{j,1}})$	$(\mathsf{sk}_0, \mathsf{sk}_1)$
$[\alpha_1]$	$(\mu \cdot \gamma_1, 0, 0, 0, \mathbf{i})$	$(\mu \cdot \gamma_1, 0, 0, 0, \mathbf{i})$	$(j, \mathsf{wit}_{z_{j,1}})$	$(\mathsf{sk}_0, \mathsf{sk}_1)$
$[\alpha_2]$	$(\mu \cdot \gamma_2, 0, 0, 0, \mathbf{i})$	$(\mu \cdot \gamma_2, 0, 0, 0, \mathbf{i})$	$(j, \mathsf{wit}_{z_{j,1}})$	$(\mathsf{sk}_0, \mathsf{sk}_1)$

Table 16: Overview of Challenge Encodings in Outer Hybrid 1

Encoding	ct_0 **encrypts**	ct_1 **encrypts**	**Witness for** π_0	**Witness for** π_1
$[\alpha_0]$	$(\mu, 0, 0, 0, \mathbf{i})$	$(\mu, 0, 0, 0, \mathbf{i})$	$(j, \mathsf{wit}_{z_{j,1}})$	$(\mathsf{sk}_0, \mathsf{sk}_1)$
$[\alpha_1]$	$(0, \mu, 0, 0, \mathbf{i})$	$(\mu \cdot \gamma_1, 0, 0, 0, \mathbf{i})$	$(j, \mathsf{wit}_{z_{j,1}})$	$(\mathsf{sk}_0, \mathsf{sk}_1)$
$[\alpha_2]$	$(0, 0, \mu, 0, \mathbf{i})$	$(\mu \cdot \gamma_2, 0, 0, 0, \mathbf{i})$	$(j, \mathsf{wit}_{z_{j,1}})$	$(\mathsf{sk}_0, \mathsf{sk}_1)$
$[\alpha_3]$	$(0, 0, 0, \mu, \mathbf{i})$	$(\mu \cdot \gamma_3, 0, 0, 0, \mathbf{i})$	$(j, \mathsf{wit}_{z_{j,1}})$	$(\mathsf{sk}_0, \mathsf{sk}_1)$

Table 17: Overview of Challenge Encodings in Outer Hybrid 2

Outer Hybrid 2. This hybrid is identical to the outer hybrid 1, except that the challenge encodings are no longer in normal form. In particular, the first FHE ciphertext in each encoding now encrypts an oblique representation of the underlying plaintext element. For ease of understanding, we explicitly lay out what the FHE ciphertexts in each of the encodings actually encrypt in Table 17. As in outer hybrid 1, each encoding also contains a NIZK proof π_0 for $z_{j,1} \in \mathcal{L}$ under the binding crs_0, and a NIZK proof π_1 for consistency under the binding crs_1. The changes from outer hybrid 1 are highlighted in red.

Outer Hybrid 3. This hybrid is identical to the outer hybrid 2, except that the challenge encodings are now entirely in oblique form. In particular, the second FHE ciphertext in each encoding now also encrypts an oblique representation of the underlying plaintext element. Each encoding continues to have a NIZK proof π_0 for $z_{j,1} \in \mathcal{L}$ under the binding crs_0, and a NIZK proof π_1 for consistency with respect to extraction under the binding crs_1. For ease of understanding, we explicitly lay out what the FHE ciphertexts in each of the encodings actually encrypt in Table 18. The changes from outer hybrid 2 are highlighted in red.

Encoding	ct_0 **encrypts**	ct_1 **encrypts**	**Witness for** π_0	**Witness for** π_1
$[\alpha_0]$	$(\mu, 0, 0, 0, \mathbf{i})$	$(\mu, 0, 0, 0, \mathbf{i})$	$(j, \mathsf{wit}_{z_{j,1}})$	$(\mathsf{sk}_0, \mathsf{sk}_1)$
$[\alpha_1]$	$(0, \mu, 0, 0, \mathbf{i})$	$(0, \mu, 0, 0, \mathbf{i})$	$(j, \mathsf{wit}_{z_{j,1}})$	$(\mathsf{sk}_0, \mathsf{sk}_1)$
$[\alpha_2]$	$(0, 0, \mu, 0, \mathbf{i})$	$(0, 0, \mu, 0, \mathbf{i})$	$(j, \mathsf{wit}_{z_{j,1}})$	$(\mathsf{sk}_0, \mathsf{sk}_1)$

Table 18: Overview of Challenge Encodings in Outer Hybrid 3

Outer Hybrid 4. In this hybrid, we make the following alterations to the manner in which the MMap is generated (again, these are essentially identical to those in outer hybrid 4 of the proof of SXDH-hardness):

1. We switch back the element $z_{j,0}$ in the public parameters from a member to a non-member for $\mathcal{L}$, i.e., we now sample $z_{j,0} \leftarrow \mathcal{X} \setminus \mathcal{L}$. The element $z_{j,1}$ continues to be a member for $\mathcal{L}$, i.e., we continue to sample $z_{j,1} \leftarrow \mathcal{L}$, along with the (unique) membership-witness $\mathsf{wit}_{z_{j,1}}$.
2. We switch the circuits for addition, inversion and multiplication as follows:

$$\widehat{\mathsf{C}}_{\mathsf{Add}} \mapsto \widehat{\widehat{\mathsf{C}}}_{\mathsf{Add}} \quad , \quad \widehat{\mathsf{C}}_{\mathsf{Inv}} \mapsto \widehat{\widehat{\mathsf{C}}}_{\mathsf{Inv}} \quad , \quad \widehat{\mathsf{C}}_{\mathsf{Mult}} \mapsto \widehat{\widehat{\mathsf{C}}}_{\mathsf{Mult}}.$$

The switched circuits for addition, inversion and multiplication are described in Figures 11, 12, and 14, respectively. Finally, note that the element y continues to be a non-member for $\mathcal{L}$ and crs_1 is still generated in the binding mode. Hence, any valid encoding must still satisfy consistency with respect to extraction.

Outer Hybrid 5. This hybrid is identical to outer hybrid 3 except that we switch the group elements g_0, g_1 and g_2 hardwired into the modified MMap circuits $\widehat{\mathsf{C}}_{\mathsf{Add}}$, $\widehat{\mathsf{C}}_{\mathsf{Inv}}$, $\widehat{\mathsf{C}}_{\mathsf{Mult}}$ and $\widehat{\mathsf{C}}_{\mathsf{ext}}$ from uniformly random to a randomly sampled k-EDDH tuple, albeit with the same base element g_0. More formally, we uniformly sample $\gamma \leftarrow \mathbb{Z}_q$, and set

$$(g_1, g_2) = (g_0^{\gamma}, g_0^{\gamma^k}).$$

Note that the final element g_3 continues to be uniformly randomly sampled.

Outer Hybrid 6. In this hybrid, we make the following alterations to the manner in which the MMap is generated (again, these are essentially identical to those in outer hybrid 6 of the proof of SXDH-hardness):

1. We switch the element $z_{j,0}$ in the public parameters from a non-member to a member for $\mathcal{L}$, i.e., we now sample $z_{j,0} \leftarrow \mathcal{L}$, along with a (unique) membership-witness $\mathsf{wit}_{z_{j,0}}$. The element $z_{j,1}$ continues to be a member for $\mathcal{L}$, i.e., we continue to sample $z_{j,1} \leftarrow \mathcal{L}$, along with the (unique) membership-witness $\mathsf{wit}_{z_{j,1}}$.
2. We switch the circuits for addition, inversion and multiplication as follows:

$$\widehat{\widehat{\mathsf{C}}}_{\mathsf{Add}} \mapsto \widehat{\mathsf{C}}_{\mathsf{Add}} \quad , \quad \widehat{\widehat{\mathsf{C}}}_{\mathsf{Inv}} \mapsto \widehat{\mathsf{C}}_{\mathsf{Inv}},$$
$$\widehat{\widehat{\mathsf{C}}}_{\mathsf{Mult}} \quad \mapsto \quad \widehat{\mathsf{C}}_{\mathsf{Mult}},$$

Outer Hybrid 7. This hybrid is identical to the outer hybrid 4, except that the challenge SXDH encodings are now switched back to a partially oblique form. In particular, the second FHE ciphertext in each encoding now encrypts a normal representation of the underlying plaintext element. Each encoding continues to have a NIZK proof π_0 for $z_{j,1} \in \mathcal{L}$ under the binding crs_0, and a NIZK proof π_1 for consistency with respect to extraction under the binding crs_1. For ease of understanding, we explicitly lay out what the FHE ciphertexts in each of the encodings actually encrypt in Table **??**. The changes from outer hybrid 4 are highlighted in red.

Outer Hybrid 8. This hybrid is identical to the outer hybrid 5, except that the challenge encodings are now switched back entirely to the normal form. In particular, the first FHE ciphertext in each encoding now also encrypts a normal representation of the underlying plaintext element. Each encoding continues to have a NIZK proof π_0 for $z_{j,1} \in \mathcal{L}$ under the binding crs_0, and a NIZK proof π_1 for consistency with respect to extraction under the binding crs_1. For ease of understanding, we explicitly lay out what the FHE ciphertexts in each of the encodings actually encrypt in Table 20. The changes from outer hybrid 5 are highlighted in red.

Encoding	ct_0 encrypts	ct_1 encrypts	Witness for π_0	Witness for π_1
$[\alpha_0]$	$(\mu, 0, 0, 0, \mathbf{i})$	$(\mu, 0, 0, 0, \mathbf{i})$	$(j, \mathsf{wit}_{z_{j,1}})$	$(\mathsf{sk}_0, \mathsf{sk}_1)$
$[\alpha_1]$	$(0, \mu, 0, 0, \mathbf{i})$	$(\mu \cdot \gamma, 0, 0, 0, \mathbf{i})$	$(j, \mathsf{wit}_{z_{j,1}})$	$(\mathsf{sk}_0, \mathsf{sk}_1)$
$[\alpha_2]$	$(0, 0, \mu, 0, \mathbf{i})$	$(\mu \cdot \gamma^k, 0, 0, 0, \mathbf{i})$	$(j, \mathsf{wit}_{z_{j,1}})$	$(\mathsf{sk}_0, \mathsf{sk}_1)$

Table 19: Overview of Challenge Encodings in Outer Hybrid 7

Encoding	ct_0 encrypts	ct_1 encrypts	Witness for π_0	Witness for π_1
$[\alpha_0]$	$(\mu, 0, 0, 0, \mathbf{i})$	$(\mu, 0, 0, 0, \mathbf{i})$	$(j, \mathsf{wit}_{z_{j,1}})$	$(\mathsf{sk}_0, \mathsf{sk}_1)$
$[\alpha_1]$	$(\mu \cdot \gamma, 0, 0, 0, \mathbf{i})$	$(\mu \cdot \gamma, 0, 0, 0, \mathbf{i})$	$(j, \mathsf{wit}_{z_{j,1}})$	$(\mathsf{sk}_0, \mathsf{sk}_1)$
$[\alpha_2]$	$(\mu \cdot \gamma^k, 0, 0, 0, \mathbf{i})$	$(\mu \cdot \gamma^k, 0, 0, 0, \mathbf{i})$	$(j, \mathsf{wit}_{z_{j,1}})$	$(\mathsf{sk}_0, \mathsf{sk}_1)$

Table 20: Overview of Challenge Encodings in Outer Hybrid 8

Outer Hybrid 9. This hybrid is identical to outer hybrid 8, except that we make the following alterations to the manner in which the MMap is set up (again, these are essentially identical to those in outer hybrid 8 of the proof of SXDH-hardness):

1. We switch back the elements $z_{j,0}$ and $z_{j,1}$ in the public parameters from members to non-members for $\mathcal{L}$, i.e., we now sample $z_{j,0}, z_{j,1} \leftarrow \mathcal{X} \setminus \mathcal{L}$. Note that this is exactly as in the real MMap scheme.

2. We switch back the circuits for addition, inversion, multiplication and extraction as follows:

$$\begin{array}{ccc} \widehat{\mathsf{C}}_{\mathsf{Add}} \mapsto \mathsf{C}_{\mathsf{Add}} & , & \widehat{\mathsf{C}}_{\mathsf{Inv}} \mapsto \mathsf{C}_{\mathsf{Inv}}, \\ \widehat{\mathsf{C}}_{\mathsf{Mult}} \mapsto \mathsf{C}_{\mathsf{Mult}} & , & \widehat{\mathsf{C}}_{\mathsf{ext}} \mapsto \mathsf{C}_{\mathsf{ext}}, \end{array}$$

   In particular, the circuits are now exactly as in the real MMap scheme, with the exception that the tuple (g_0, g_1, g_2) hardwired inside these circuits continues to be a k-EDDH tuple (and the corresponding secret exponents hardwired into the multiplication circuit are γ and γ^k).

Additionally, we make the following change to the manner in which the challenge encodings are generated: for each encoding, the NIZK proof π_0 under the binding crs_0 now proves that the encoding is in the normal representation using the witness $(\mathsf{sk}_0, \mathsf{sk}_1)$, as opposed to proving that $z_{j,1} \in \mathcal{L}$. For ease of understanding, we explicitly lay out what the FHE ciphertexts in each of the encodings actually encrypt in Table 21. The changes from outer hybrid 8 are highlighted in red.

Outer Hybrid 10. This hybrid is identical to outer hybrid 7 except that we switch the tuple of group elements (g_0, g_1, g_2) hardwired into the MMap circuits $\mathsf{C}_{\mathsf{Add}}$, $\mathsf{C}_{\mathsf{Inv}}$, $\mathsf{C}_{\mathsf{Mult}}$ and $\mathsf{C}_{\mathsf{ext}}$ from a uniform k-EDDH tuple to a uniformly random tuple, albeit with the same base element g_0 (and the corresponding secret exponents hardwired into the multiplication circuit are switched from γ and γ^k to uniformly random). In other words, the MMap circuits are now exactly as in the real scheme.

Encoding	ct_0 **encrypts**	ct_1 **encrypts**	**Witness for** π_0	**Witness for** π_1
$[\alpha_0]$	$(\mu, 0, 0, 0, \mathbf{i})$	$(\mu, 0, 0, 0, \mathbf{i})$	$(\mathsf{sk}_0, \mathsf{sk}_1)$	$(\mathsf{sk}_0, \mathsf{sk}_1)$
$[\alpha_1]$	$(\mu \cdot \gamma, 0, 0, 0, \mathbf{i})$	$(\mu \cdot \gamma, 0, 0, 0, \mathbf{i})$	$(\mathsf{sk}_0, \mathsf{sk}_1)$	$(\mathsf{sk}_0, \mathsf{sk}_1)$
$[\alpha_2]$	$(\mu \cdot \gamma^k, 0, 0, 0, \mathbf{i})$	$(\mu \cdot \gamma^k, 0, 0, 0, \mathbf{i})$	$(\mathsf{sk}_0, \mathsf{sk}_1)$	$(\mathsf{sk}_0, \mathsf{sk}_1)$

Table 21: Overview of Challenge Encodings in Outer Hybrid 9

7.2 Indistinguishability of Outer Hybrids

In this section, we argue that the outer hybrids are computationally indistinguishable from each other. A majority of the arguments are very similar to those used in the proof of Theorem 6.1, and are hence not detailed. We expand on the arguments that are specific to k-EDDH.

Outer Hybrids 0 and 1. We first argue that outer hybrids 0 and 1 are computationally indistinguishable from each other. In particular, we state and prove the following lemma:

Lemma 7.2. *The outer hybrids 0 and 1 are computationally indistinguishable provided that all of the following assumptions hold:*

- *The piO scheme is indistinguishability-secure against X-IND samplers.*
- *Membership in $\mathcal{L}$ is computationally hard to decide,*
- *Members of $\mathcal{L}$ have unique witnesses for membership.*
- *The dual-mode NIZK proof system satisfies CRS indistinguishability, perfect extractability in the binding mode, and perfect witness-indistinguishability in the hiding mode.*

Proof. The proof of this lemma uses a sequence of inner hybrids that are essentially identical to those used in the proof of Lemma 6.2. Hence, we do not detail them.

Outer Hybrids 1 and 2. We now argue that outer hybrids 1 and 2 are computationally indistinguishable from each other. In particular, we state and prove the following lemma:

Lemma 7.3. *The outer hybrids 1 and 2 are computationally indistinguishable provided that all of the following assumptions hold:*

- *The piO scheme is indistinguishability-secure against X-IND samplers.*
- *Membership in $\mathcal{L}$ is computationally hard to decide,*
- *Members of $\mathcal{L}$ have unique witnesses for membership.*
- *The dual-mode NIZK proof system satisfies CRS indistinguishability, perfect extractability in the binding mode, and perfect witness-indistinguishability in the hiding mode.*

Proof. The proof of this lemma again uses a sequence of inner hybrids that are essentially identical to those used in the proof of Lemma 6.3. Hence, we do not detail them.

Outer Hybrids 2 and 3. We now argue that outer hybrids 2 and 3 are computationally indistinguishable from each other. In particular, we state and prove the following lemma:

Lemma 7.4. *The outer hybrids 2 and 3 are computationally indistinguishable provided that all of the following assumptions hold:*

- *The piO scheme is indistinguishability-secure against X-IND samplers.*
- *Membership in $\mathcal{L}$ is computationally hard to decide,*
- *Members of $\mathcal{L}$ have unique witnesses for membership.*
- *The dual-mode NIZK proof system satisfies CRS indistinguishability, perfect extractability in the binding mode, and perfect witness-indistinguishability in the hiding mode.*

Proof. To prove this lemma, we use another sequence of inner hybrids that are essentially identical to those used in the proof of Lemma 6.6. Hence, we do not detail them. □

Outer Hybrids 3 and 4. We state and prove the following lemma:

Lemma 7.5. *The outer hybrids 3 and 4 are computationally indistinguishable provided that all of the following assumptions hold:*

- *The piO scheme is indistinguishability-secure against X-IND samplers.*
- *Membership in $\mathcal{L}$ is computationally hard to decide,*
- *Members of $\mathcal{L}$ have unique witnesses for membership.*
- *The dual-mode NIZK proof system satisfies CRS indistinguishability, perfect extractability in the binding mode, and perfect witness-indistinguishability in the hiding mode.*

Outer Hybrids 4 and 5. We state and prove the following lemma:

Lemma 7.6. *The outer hybrids 3 and 4 are computationally indistinguishable provided that the k-EDDH assumption holds over the group $\mathbb{G}$.*

Proof. Suppose that there exists a PPT adversary $\mathcal{A}$ that can distinguish between the outer hybrids 3 and 4 with non-negligible probability. We construct a PPT algorithm $\mathcal{B}$ that can break the k-EDDH assumption over the group $\mathbb{G}$ with non-negligible probability. As part of its input k-EDDH challenge, $\mathcal{B}$ receives a tuple of group elements of the form (g_0, g_1, g_2), and sets up the public parameters for the MMap as follows:

1. $\mathcal{B}$ samples two key-pairs for the FHE scheme as:

$$(\mathsf{pk}_0, \mathsf{sk}_0), (\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathsf{FHE.Gen}(1^\lambda),$$

   and a pair of binding NIZK CRS strings (along with the corresponding extraction trapdoors) as:

$$(\mathsf{crs}_0, \mathsf{t}_{\mathsf{ext},0}), (\mathsf{crs}_1, \mathsf{t}_{\mathsf{ext},1}) \leftarrow \mathsf{NIZK.Setup}(1^\lambda, \mathsf{binding}).$$

2. Next $\mathcal{B}$ uniformly samples a total $(n+1)$ elements from the set $\mathcal{X}$ that are all non-members for the subset $\mathcal{L}$ and one element that is a member for $\mathcal{L}$. More formally, it samples

$$y, z_{1,0}, z_{1,1}, \ldots, z_{j-1,0}, z_{j-1,1}, z_{j,0}, z_{j+1,0}, z_{j+1,1} \ldots, z_{n,0}, z_{n,1} \leftarrow \mathcal{X} \setminus \mathcal{L},$$

$$z_{j,1} \leftarrow \mathcal{L},$$

   where $z_{j,1}$ has unique membership-witness $\mathsf{wit}_{z_{j,1}}$.

3. Finally, $\mathcal{B}$ sets up the MMap circuits $\widehat{\widehat{\widehat{\mathsf{C}}}}_{\mathsf{Add}}$, $\widehat{\widehat{\widehat{\mathsf{C}}}}_{\mathsf{Inv}}$, $\widehat{\widehat{\widehat{\mathsf{C}}}}_{\mathsf{Mult}}$ and $\widehat{\mathsf{C}}_{\mathsf{ext}}$ exactly as described in Figures 11, 12, 14 and 10, respectively, except for the fact that it hardwires the tuple (g_0, g_1, g_2, g_3) into each of these circuits, where g_0, g_1 and g_2 are part of the input challenge, and g_3 is uniformly sampled. It then computes and outputs four probabilistically indistinguishability-obfuscated circuits of the form

$$\bar{\mathsf{C}}_{\mathsf{Add}} = \mathsf{piO.Obf}(\widehat{\mathsf{C}}_{\mathsf{Add}}) \quad , \quad \bar{\mathsf{C}}_{\mathsf{Inv}} = \mathsf{piO.Obf}(\widehat{\mathsf{C}}_{\mathsf{Inv}}),$$
$$\bar{\mathsf{C}}_{\mathsf{Mult}} = \mathsf{piO.Obf}(\widehat{\mathsf{C}}_{\mathsf{Mult}}) \quad , \quad \bar{\mathsf{C}}_{\mathsf{ext}} = \mathsf{piO.Obf}(\widehat{\mathsf{C}}_{\mathsf{ext}}).$$

Next, $\mathcal{B}$ sets up the challenge encodings in the oblique representation as follows (refer to Table 18 for how the challenge encodings are formatted in oblique representation in outer hybrid 3):

1. $\mathcal{B}$ samples $\mu \leftarrow \mathbb{Z}_q$ and generates the following FHE ciphertexts for each $b \in \{0, 1\}$:

$$\mathsf{ct}_{b,0} = \mathsf{FHE.Enc}(\mathsf{pk}_b, (\mu, 0, 0, 0, \mathbf{i})) \quad , \quad \mathsf{ct}_{b,1} = \mathsf{FHE.Enc}(\mathsf{pk}_b, (0, \mu, 0, 0, \mathbf{i})),$$
$$\mathsf{ct}_{b,2} = \mathsf{FHE.Enc}(\mathsf{pk}_b, (0, 0, \mu, 0, \mathbf{i})).$$

2. $\mathcal{B}$ generates the following proofs for each $\ell \in [0, 2]$:

$$\begin{aligned} \pi_{0,\ell} &= \mathsf{NIZK.Prove}(\mathsf{crs}_0, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_{0,\ell}, \mathsf{ct}_{1,\ell}, \mathbf{i}, \{z_{j,1}\}_{j \in [0,n]}), (j, \mathsf{wit}_{z_{j,1}})), \\ \pi_{1,\ell} &= \mathsf{NIZK.Prove}(\mathsf{crs}_1, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_{0,\ell}, \mathsf{ct}_{1,\ell}, \mathbf{i}, y), (\mathsf{sk}_0, \mathsf{sk}_1)). \end{aligned}$$

3. Finally, for each $\ell \in [0, 2]$, $\mathcal{B}$ generates the encoding $[\alpha_\ell]$ as:

$$[\alpha_\ell] = \Big(\mathsf{ct}_{0,\ell}, \mathsf{ct}_{1,\ell}, \mathbf{i}, \pi_{0,\ell}, \pi_{1,\ell}\Big).$$

$\mathcal{B}$ then provides $\mathcal{A}$ with the MMap public parameters and the challenge encodings. Eventually, $\mathcal{A}$ outputs a bit $b^\star$. $\mathcal{B}$ outputs the same bit $b^\star$. Now, observe the following:

- Suppose that $\mathcal{B}$ receives as input a tuple of uniformly random group elements of the form $(g_0, g_0^{\gamma_1}, g_0^{\gamma_2})$. In this case, the view of $\mathcal{A}$ is exactly as in outer hybrid 4.
- On the other hand, when $\mathcal{B}$ receives as input a DDH tuple of the form $(g_0, g_0^{\gamma}, g^{\gamma^k})$, the view of $\mathcal{A}$ is exactly as in outer hybrid 5. □

Hence the advantage of $\mathcal{B}$ in breaking k-EDDH over the group $\mathbb{G}$ is the same as the advantage of $\mathcal{A}$ in distinguishing the outer hybrids 4 and 5. This concludes the proof of Lemma 7.6.

Outer Hybrids 5 and 6. We now argue that outer hybrids 5 and 6 are computationally indistinguishable from each other. In particular, we state and prove the following lemma:

Lemma 7.7. *The outer hybrids 5 and 6 are computationally indistinguishable provided that all of the following assumptions hold:*

- *The piO scheme is indistinguishability-secure against X-IND samplers.*
- *Membership in $\mathcal{L}$ is computationally hard to decide,*
- *Members of $\mathcal{L}$ have unique witnesses for membership.*
- *The dual-mode NIZK proof system satisfies CRS indistinguishability, perfect extractability in the binding mode, and perfect witness-indistinguishability in the hiding mode.*

Proof. The proof of this lemma uses essentially the same inner hybrids as in the proof of Lemma 7.5, albeit in reverse order, and is hence not detailed. □

Outer Hybrids 6 and 7. We state and prove the following lemma:

Lemma 7.8. *The outer hybrids 6 and 7 are computationally indistinguishable provided that all of the following assumptions hold:*

- *The piO scheme is indistinguishability-secure against X-IND samplers.*
- *Membership in $\mathcal{L}$ is computationally hard to decide,*
- *Members of $\mathcal{L}$ have unique witnesses for membership.*
- *The dual-mode NIZK proof system satisfies CRS indistinguishability, perfect extractability in the binding mode, and perfect witness-indistinguishability in the hiding mode.*

Proof. The proof of this lemma uses essentially the same inner hybrids as in the proof of Lemma 7.4, albeit in reverse order, and is hence not detailed. □

Outer Hybrids 7 and 8. We state the following lemma:

Lemma 7.9. *The outer hybrids 7 and 8 are computationally indistinguishable provided that all of the following assumptions hold:*

- *The piO scheme is indistinguishability-secure against X-IND samplers.*
- *Membership in $\mathcal{L}$ is computationally hard to decide,*
- *Members of $\mathcal{L}$ have unique witnesses for membership.*
- *The dual-mode NIZK proof system satisfies CRS indistinguishability, perfect extractability in the binding mode, and perfect witness-indistinguishability in the hiding mode.*

Proof. The proof of this lemma uses essentially the same inner hybrids as in the proof of Lemma 7.3, albeit in reverse order, and is hence not detailed. □

Outer Hybrids 8 and 9. We state the following lemma:

Lemma 7.10. *The outer hybrids 8 and 9 are computationally indistinguishable provided that all of the following assumptions hold:*

- *The piO scheme is indistinguishability-secure against X-IND samplers.*
- *Membership in $\mathcal{L}$ is computationally hard to decide,*
- *Members of $\mathcal{L}$ have unique witnesses for membership.*
- *The dual-mode NIZK proof system satisfies CRS indistinguishability, perfect extractability in the binding mode, and perfect witness-indistinguishability in the hiding mode*

Proof. The proof of this lemma uses essentially the same inner hybrids as in the proof of Lemma 7.2, albeit in reverse order, and is hence not detailed. □

Outer Hybrids 9 and 10. We state and prove the following lemma:

Lemma 7.11. *The outer hybrids 9 and 10 are computationally indistinguishable provided that all of the following assumptions hold:*

- *The piO scheme is indistinguishability-secure against X-IND samplers.*
- *The dual-mode NIZK proof system satisfies perfect soundness in the binding mode.*

Proof. In this hybrid, we switch the tuple of group elements (g_0, g_1, g_2, g_3) hardwired into the MMap circuits as follows: we switch the the tuple of exponents $(\gamma_1, \gamma_2, \gamma_3)$ from a triplet of the form $(\gamma, \gamma^k, \gamma_3)$ to a uniformly random triple, i.e., γ_2 is now sampled uniformly at random from $\mathbb{Z}_q$. Effectively, this switches the sub-tuple of group elements (g_0, g_1, g_2) from a uniform k-EDDH tuple to a uniformly random tuple, albeit with the same base element g_0.

We argue below that this switch does not alter the output distribution of these circuits from outer hybrid 9 to outer hybrid 10. Once this is established, the indistinguishability argument follows immediately under the assumption that the piO scheme is indistinguishability-secure against X-IND samplers (once for each MMap circuit). The argument is very similar to that presented in the proof of Lemma 6.13, but is presented for the sake of completeness.

We first focus on the MMap circuits $\mathsf{C}_{\mathsf{Add}}$ and $\mathsf{C}_{\mathsf{Inv}}$. Observe that switching (g_0, g_1, g_2) from a uniform k-EDDH tuple to a uniformly random tuple only (potentially) affects the outcome of the "**If**" condition in step 7 of each of these circuits. Note however that in both outer hybrids 9 and 10, crs_0 is in binding mode, each $z_{j,b}$ in the public parameter is a non-member for $j \in [n]$ and $b \in \{0, 1\}$, and y in the public parameter is also a non-member. Hence, it follows from the perfect soundness of the dual-mode NIZK proof system that any input encoding(s) for which these circuits do not output $\perp$ and terminate before step 7 must be *both* in normal representation *and* consistent. This in turn guarantees that the outcome of the "**If**" condition in step 7 must be "false". Hence, the output distributions of the MMap circuits $\mathsf{C}_{\mathsf{Add}}$ and $\mathsf{C}_{\mathsf{Inv}}$ in outer hybrids 9 and 10 are identical.

Next, we focus on the MMap circuit $\mathsf{C}_{\mathsf{Mult}}$. Observe that switching (g_0, g_1, g_2) from a uniform k-EDDH tuple to a uniformly random tuple only (potentially) affects the outcome of the "**Else If**" condition in step 3 of the circuit $\mathsf{C}_{\mathsf{Mult,FHE}}$ (Figure 4), which is evaluated homomorphically in Step 5 of $\mathsf{C}_{\mathsf{Mult}}$. Note however that in both outer hybrids 9 and 10, crs_0 is in binding mode, each $z_{j,b}$ in the public parameter is a non-member for $j \in [n]$ and $b \in \{0, 1\}$, and y in the public parameter is also a non-member. Hence, it follows from the perfect soundness of the dual-mode NIZK proof system that any input encoding(s) for which these circuits do not output $\perp$ and terminate before step 7 must be *both* in normal representation *and* consistent. Hence, the "**Else If**" condition in step 3 of the circuit $\mathsf{C}_{\mathsf{Mult,FHE}}$ can never be satisfied by a pair of valid input encodings. This in turn implies that the output distributions of the MMap circuit $\mathsf{C}_{\mathsf{Mult}}$ in outer hybrids 9 and 10 are identical.

Finally, we focus on the MMap extraction circuit $\mathsf{C}_{\mathsf{ext}}$. Observe that switching (g_0, g_1, g_2) from a uniform k-EDDH tuple to a uniformly random tuple potentially affects the output of the extraction circuit. However, note yet again that in both outer hybrids 9 and 10, crs_0 is in binding mode, each $z_{j,b}$ in the public parameter is a non-member for $j \in [n]$ and $b \in \{0, 1\}$, and y in the public parameter is also a non-member. Hence, it follows from the perfect soundness of the dual-mode NIZK proof system that any input encoding(s) for which the circuit does not output $\perp$ and terminate before the extraction step is executed must be *both* in normal representation *and* consistent. Observe also that the base element g_0 in the tuple of group elements remains unaltered across outer hybrids 9 and 10. It follows immediately that the outcome of extraction on a given encoding in normal representation in both hybrids is identical. In other words, the output distributions of $\mathsf{C}_{\mathsf{ext}}$ in outer hybrids 9 and 10 are identical.

This concludes the proof of Lemma 7.11, and hence the proof of Theorem 7.1.

8 Achieving Joint-SXDH Hardness

In this section, we demonstrate that the asymmetric MMap construction can be upgraded to achieve non-adaptive joint-SXDH hardness.

Normal v/s Oblique Encodings. Recall that in our SXDH-hard (and k-EDDH-hard) asymmetric MMap construction, an encoding could be in one of two forms - normal or oblique, depending on the plaintexts underlying the FHE ciphertexts. To briefly recap, suppose that the plaintexts underlying the FHE ciphertexts in a given encoding at level $\mathbf{i}$ are of the form:

$$(a_{0,0}, a_{1,0}, a_{2,0}, a_{3,0}, \mathbf{i}) \quad , \quad (a_{0,1}, a_{1,1}, a_{2,1}, a_{3,1}, \mathbf{i}).$$

For $b \in \{0, 1\}$, the tuple $(a_{0,b}, a_{1,b}, a_{2,b}, a_{3,b}, \mathbf{i})$ is said to be in "normal form" if

$$(a_{0,b}, a_{1,b}, a_{2,b}, a_{3,b}, \mathbf{i}) = (a, 0, 0, 0, \mathbf{i})$$

for some $a \in \mathbb{Z}_q$. Otherwise, it is said to be in "oblique-form". Depending on the forms of the tuples underlying the FHE ciphertexts, we classify an encoding into one of three representations:

- **Normal representation:** Both tuples are in normal form.
- **Partially oblique representation:** Exactly one of the tuples is in normal form, while the other is in oblique form. Unless otherwise specified, we assume that the second tuple is in normal form.
- **Oblique representation:** Both tuples are in oblique form.

Our proof of SXDH (and that of k-EDDH) uses the oblique representation of the challenge encodings in an essential way, but enabling oblique representation of the encodings in the challenge level-set needs to be done in a careful manner. In particular, in certain hybrids of the proof, oblique representation cannot be enabled in any given level-set because we cannot efficiently compute the oblique representation for the product two encodings that are both in oblique representation unless we know the secret exponents underlying the tuple of group elements (g_0, g_1, g_2, g_3). This restriction arises from our extraction procedure. Given an oblique tuple of the form $(a_{0,0}, a_{1,0}, a_{2,0}, a_{3,0}, \mathbf{i})$, the extraction procedure computes

$$g^* = \prod_{\ell=0}^{\ell=3} g_\ell^{a_{\ell,0}}.$$

Clearly, in order to multiply a pair of encodings in oblique form, we would need to know the discrete log of the hardwired group elements (g_1, g_2, g_3) with respect to the public group element g; however in certain hybrids of the proofs of SXDH and k-EDDH (in particular, the outer hybrids 4 and 5 in Tables 1 and 23), the proof must explicitly avoid the knowledge of these exponents when creating the obfuscated multiplication circuit. Hence, in these specific hybrids, when we enable oblique encodings, we ensure that any pair of level-sets that are allowed to support oblique encodings must be incompatible for multiplication. To achieve this, we use a language-membership "trapdoor" to enforce that any level set $\mathbf{i}'$ (including the challenge level-set $\mathbf{i}$ chosen by the adversary) is allowed to support oblique encodings if and only if $\mathbf{i}'_j = 1$ for a fixed j.

Challenges for Joint-SXDH. Unfortunately, the same approach does not work when proving non-adaptive joint-SXDH. For instance, suppose we have a 4-linear map (i.e. $n = 4$), and suppose that in the non-adaptive joint-SXDH game, the adversary commits to the following set $T = (\mathbf{i}_1, \mathbf{i}_2, \mathbf{i}_3)$ of level-sets:

$$\mathbf{i}_1 = \begin{bmatrix} 0 & 1 & 1 & 0 \end{bmatrix} \quad , \quad \mathbf{i}_2 = \begin{bmatrix} 0 & 0 & 1 & 1 \end{bmatrix} \quad , \quad \mathbf{i}_3 = \begin{bmatrix} 0 & 1 & 0 & 1 \end{bmatrix}.$$

In order to use the same strategy as in our proofs of SXDH and k-EDDH, we would ideally want to allow oblique encodings in each of these level sets. Unfortunately, there is no $j \in [n]$ such that all of these challenge level-sets subsume the j^{th} unit level-set.

A Different Restriction for Oblique Encodings. To tackle this issue, we introduce a different restriction for level-sets supporting oblique representation. For simplicity, we first show how the restriction would have worked in the proof of SXDH (or k-EDDH) where the adversary only chooses a single challenge level-set. Subsequently, we show how this extends to the proof of joint-SXDH.

Suppose that we enforce that any level set $\mathbf{i}'$ is allowed to support oblique encodings if and only if it subsumes the challenge level-set. Note that assuming we can somehow efficiently enforce this restriction, it equivalently achieves the same desired effects as the previous restriction. It allows us to switch between normal and oblique encodings for the challenge level-set. It also ensures that any pair of level-sets that are allowed to support oblique encodings must be incompatible for multiplication.

At this point, the question is: how do we achieve this restriction in our asymmetric MMap construction? The answer is: we simply use *multiple* language-membership trapdoors as opposed to a single language-membership trapdoor. Let $z_{1,0}, z_{1,1}, \ldots, z_{n,0}, z_{n,1}$ be the set of elements in the public parameter of the asymmetric MMap construction. We change the statement for the proof π_0 in each encoding at level $\mathbf{i}'$ as follows: π_0 is now a verifying NIZK proof of "normal representation" that (informally) proves under crs_0 that one of the following statements must be true:

1. Either the encoding is in the normal representation.

2. Or there exists a $b^* \in \{0,1\}$ such that for each $j \in [n]$ where $\mathbf{i}'_j = b^*$, we have $z_{j,b^*} \in \mathcal{L}$.

We now show how this helps in the proof of security. Let $\mathbf{i}$ be the challenge level-set chosen by the adversary. When generating the public parameters of our scheme in the various hybrids in the proof of SXDH (and k-EDDH), we use the following strategy:

- In any hybrid where earlier we had both $z_{j,0}$ and $z_{j,1}$ as non-members of the language $\mathcal{L}$ (outer hybrids 0, 9 and 10 in Tables 1 and 23), we will now have $z_{j,b}$ for each $j \in [n]$ and $b \in \{0,1\}$ as a non-member of the language $\mathcal{L}$.

- In any hybrid where earlier we had $z_{j,0}$ as a non-member and $z_{j,1}$ as a member of the language $\mathcal{L}$ (outer hybrids 4 and 5 in Tables 1 and 23), we will now have $z_{j,0}$ for each $j \in [n]$ such that $\mathbf{i}_j = 0$ as a member of the language $\mathcal{L}$ ($\mathbf{i}$ being the challenge level-set in the SXDH/k-EDDH game). All other elements will be sampled as non-members of the language $\mathcal{L}$.

- In any hybrid where earlier we had both $z_{j,0}$ and $z_{j,1}$ as members of the language $\mathcal{L}$ (all remaining outer hybrids in Tables 1 and 23), we will now have $z_{j,b}$ for each $j \in [n]$ and $b \in \{0,1\}$ as a member of the language $\mathcal{L}$.

Now, consider an outer hybrid where $z_{j,0}$ for each $j \in [n]$ such that $\mathbf{i}_j = 0$ is a member of the language $\mathcal{L}$ ($\mathbf{i}$ being the challenge level-set in the non-adaptive SXDH/k-EDDH game), while all other elements are non-members. This "trapdoor" allows not only the challenge level-set to support valid oblique encodings, but any level-set that subsumes the challenge level-set (because for such each level-set, we can use a subset of the available membership witnesses). On the other hand, any level-set that does not entirely subsume the challenge-level set is disallowed from supporting oblique encodings, since it would require producing a membership witness for some $z_{j,b} \notin \mathcal{L}$.

Extension to Joint-SXDH. The only remaining item to address is how this approach can be extended for non-adaptive joint-SXDH. At a high level, in the appropriate hybrids in the security proof, we relax the restriction on encodings in oblique representation as follows: any level set $\mathbf{i}'$ is allowed to support oblique encodings if and only if it subsumes at least one of the challenge level-sets. Note that assuming we can somehow efficiently enforce this restriction, it effectively achieves the same desired effects. It allows us to switch between normal and oblique encodings for the challenge level-set. In addition, since any pair of challenge level-sets must be incompatible with respect to multiplication (from the definition of joint-SXDH), the restriction also ensures that any pair of level-sets that are allowed to support oblique encodings must be incompatible for multiplication, and hence multiplication of two valid input-encodings does not require the knowledge of the secret exponents.

To achieve this restriction in our asymmetric MMap construction, we simply use *multiple sets of* language-membership trapdoors as opposed to a single set of language-membership trapdoors. Concretely, we use two matrices $\mathbf{Z}_0$ and $\mathbf{Z}_1$ of group elements in the public parameter , where for each $b \in \{0,1\}$, we have

$$\mathbf{Z}_b = \begin{bmatrix} z_{1,1,b} & z_{1,2,b} & \cdots & z_{1,n,b} \\ \vdots & \vdots & \ddots & \vdots \\ z_{|T|,1,b} & z_{|T|,2,b} & \cdots & z_{|T|,n,b} \end{bmatrix}$$

where $|T|$ is a parameter denoting the size of the set T of challenge level-sets chosen by the adversary in the non-adaptive joint-SXDH game. This means that our construction is now parameterized by the joint-SXDH assumptions that it supports, but for most known applications for MMaps with joint-SXDH, this parameter is apriori bounded. We further change the statement for the proof π_0 in each encoding at level $\mathbf{i}'$ as follows: π_0 is now a verifying NIZK proof of "normal representation" that (informally) proves under crs_0 that one of the following statements must be true:

1. Either the encoding is in the normal representation.
2. Or there exists some $t \in [|T|]$ and some $b^* \in \{0,1\}$ such that for each $j \in [n]$ where $\mathbf{i}'_j = b^*$, we have $z_{t,j,b^*} \in \mathcal{L}$.

We now show how this helps in the proof of security. Let $T = \{\mathbf{i}_1, \ldots, \mathbf{i}_{|T|}\}$ be the challenge level-sets chosen by the adversary. In the appropriate hybrid of the proof, for each $t \in [|T|]$ we specifically switch each element $z_{t,j,0}$ in the public parameter such that $\mathbf{i}_{t,j} = 0$ from a non-member to a member. This "trapdoor" allows not only all the challenge level-sets to support oblique encodings, but any level-set that subsumes at least one of the challenge level-set (because for such each level-set, we can use a subset of the available membership witnesses). On the other hand, any level-set that does not entirely subsume any challenge-level set is disallowed from supporting oblique encodings, since it would require producing a membership witness for some $z_{t,j'} \notin \mathcal{L}$.

8.1 The Construction

For completeness, we describe the construction of the joint-SXDH-hard asymmetric MMap given the following cryptoprimitives:

- A probabilistic-iO scheme piO $= (\mathsf{piO.Obf}, \mathsf{piOEval})$.
- A fully-homomorphic encryption scheme

$$\text{FHE} = (\mathsf{FHE.Gen}, \mathsf{FHE.Enc}, \mathsf{FHE.Dec}, \mathsf{FHE.Eval}),$$

  such that the message space is $\mathbb{Z}_q$ for some prime $q = \mathrm{poly}(\lambda)$ (λ being the security parameter).
- A dual mode NIZK argument system NIZK $= (\mathsf{NIZK.Setup}, \mathsf{NIZK.Prove}, \mathsf{NIZK.Verify})$ that is:
  - perfectly sound and extractable in the binding mode, and
  - perfectly witness indistinguishable and perfectly zero-knowledge in the hiding mode.
- 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 $y \in \mathcal{L}$.
- A pairing-free group $\mathbb{G}$ of prime order q.

8.2 Setup

The setup algorithm for our MMap construction takes as input the security parameter 1^λ, a second parameter 1^n for the degree of multilinearity, and a third parameter $1^{|T|}$ denoting the size of the set T of challenge level-sets chosen by the adversary in the non-adaptive joint-SXDH game. It samples two key-pairs for the FHE scheme as:

$$(\mathsf{pk}_0, \mathsf{sk}_0), (\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathsf{FHE.Gen}(1^\lambda),$$

and a tuple of four uniformly random group elements from the group $\mathbb{G}$ as:

$$g_0, g_1, g_2, g_3 \leftarrow \mathbb{G}.$$

It also samples a pair of binding NIZK CRS strings (along with the corresponding extraction trapdoors) as:

$$(\mathsf{crs}_0, \mathsf{t}_{\mathsf{ext},0}), (\mathsf{crs}_1, \mathsf{t}_{\mathsf{ext},1}) \leftarrow \mathsf{NIZK.Setup}(1^\lambda, \mathsf{binding}).$$

The languages for which statements are proven under these CRS strings are described subsequently.

Next the setup algorithm uniformly samples a total $(2|T| \cdot n + 1)$ elements from the set $\mathcal{X}$ that are all non-members for the subset $\mathcal{L}$. More formally, it samples $y \leftarrow \mathcal{X} \setminus \mathcal{L}$ and for each $(t, b) \in [|T|] \times \{0, 1\}$, the following elements:

$$z_{t,1,b}, z_{t,2,b}, \ldots, z_{t,n,b} \leftarrow \mathcal{X} \setminus \mathcal{L}.$$

Finally, the setup algorithm computes and outputs four probabilistically indistinguishability-obfuscated circuits of the form

$$\bar{\mathsf{C}}_{\mathsf{Add}} = \mathsf{piO.Obf}(\mathsf{C}_{\mathsf{Add}}) \quad , \quad \bar{\mathsf{C}}_{\mathsf{Inv}} = \mathsf{piO.Obf}(\mathsf{C}_{\mathsf{Inv}}),$$
$$\bar{\mathsf{C}}_{\mathsf{Mult}} = \mathsf{piO.Obf}(\mathsf{C}_{\mathsf{Mult}}) \quad , \quad \bar{\mathsf{C}}_{\mathsf{ext}} = \mathsf{piO.Obf}(\mathsf{C}_{\mathsf{ext}})$$

where $\bar{\mathsf{C}}_{\mathsf{Add}}$, $\bar{\mathsf{C}}_{\mathsf{Inv}}$ $\bar{\mathsf{C}}_{\mathsf{Mult}}$, and $\bar{\mathsf{C}}_{\mathsf{ext}}$ are the circuits for adding, inverting, multiplying, and extracting from encodings at any given "non-zero" level.

8.3 Encodings

Level-i Encodings. We describe the procedure of encoding a plaintext element at any "non-zero" encoding level $\mathbf{i}$ such that $\vec{0} < \mathbf{i} \leq \vec{1}_n$ (the encoding for a level-zero vector is the same as in Appendix 5). An encoding of a plaintext element $a \in \mathbb{Z}_q$ at level-vector $\mathbf{i}$ is a tuple of the form $(\mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, \pi_0, \pi_1)$, where:

- ct_0 and ct_1 are FHE encryptions of a tuple of the form:

$$(a_{0,0}, a_{1,0}, a_{2,0}, a_{3,0}, \mathbf{i}) \quad , \quad (a_{0,1}, a_{1,1}, a_{2,1}, a_{3,1}, \mathbf{i}).$$

  under the public key-secret key pairs $(\mathsf{pk}_0, \mathsf{sk}_0)$ and $(\mathsf{pk}_1, \mathsf{sk}_1)$ respectively.

- π_0 is a verifying NIZK proof of "normal representation". Informally, it proves under crs_0 that one of the following statements must be true:

  1. Either $a_{0,0} = a_{0,1} = a$ and $a_{\ell,0} = a_{\ell,1} = 0$ for $\ell \in \{1, 2, 3\}$.
  2. Or there exists some $t \in [|T|]$ and some $b^* \in \{0, 1\}$ such that for each $j \in [n]$ where $\mathbf{i}'_j = b^*$, we have $z_{t,j,b^*} \in \mathcal{L}$.

  Formally, π_0 is a verifying NIZK proof under crs_0 of the **OR** relation R_0 defined below ($\mathcal{K}_{\mathrm{FHE}}$ being the set of all valid key pairs under the FHE scheme):

Relation $\mathsf{R}_{\mathcal{L}}$:

$\mathsf{R}_{\mathcal{L}}(z, \mathsf{wit}_z) = 1$ if and only if $z \in \mathcal{L}$ with membership witness wit_z.

Relation $\mathsf{R}_{0,0}$:

$\mathsf{R}_{0,0}((\mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, \mathsf{pk}_0, \mathsf{pk}_1), \mathsf{wit}) = 1$ if and only if:

- **EITHER** $\mathsf{wit} = (\mathsf{sk}_0, \mathsf{sk}_1)$ *and* $(\mathsf{pk}_0, \mathsf{sk}_0), (\mathsf{pk}_1, \mathsf{sk}_1) \in \mathcal{K}_{\mathrm{FHE}}$ *and*

$$\mathsf{FHE.Dec}(\mathsf{sk}_0, \mathsf{ct}_0) = \mathsf{FHE.Dec}(\mathsf{sk}_1, \mathsf{ct}_1) = (a, 0, 0, 0, \mathbf{i}) \text{ for some } a \in \mathbb{Z}_q.$$

- **OR** $\mathsf{wit} = (a, \mathsf{r}_0, \mathsf{r}_1)$ for some $a \in \mathbb{Z}_q$ *and* for each $b \in \{0, 1\}$, we have:

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

Relation $\mathsf{R}_{0,1}$:

$\mathsf{R}_{0,1}(\{z_{t,j,b}\}_{t\in[|T|]j\in[n],b\in\{0,1\}}, \mathbf{i}, \mathsf{wit}) = 1$ if and only if:

- $\mathsf{wit} = (t, \mathcal{J}, \{\mathsf{wit}_{z,j}\}_{j\in\mathcal{J}})$ for some $\mathcal{J} \in \mathcal{P}([n])$ *and* for each $j \in \mathcal{J}$, we have $\mathsf{R}_{\mathcal{L}}(z_{t,j,\mathbf{i}_j}, \mathsf{wit}_{z,j}) = 1$.

Relation R_0:

$\mathsf{R}_0((\mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, \mathsf{pk}_0, \mathsf{pk}_1, \{z_{t,j,b}\}_{t\in[|T|]j\in[n],b\in\{0,1\}}), \mathsf{wit}) = 1$ if and only if:

- **EITHER** $\mathsf{R}_{0,0}((\mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, \mathsf{pk}_0, \mathsf{pk}_1), \mathsf{wit}) = 1$.
- **OR** $\mathsf{R}_{0,1}(\{z_{t,j,b}\}_{t\in[|T|]j\in[n],b\in\{0,1\}}, \mathbf{i}, \mathsf{wit}) = 1$.

- π_1 is a verifying NIZK proof of "consistency" that holds irrespective of whether the encoding is in normal representation or (partially) oblique representation. Informally, it proves under crs_1 that one of the following statements must be true with respect to the tuple of group elements (g_0, g_1, g_2, g_3) embedded inside the MMap circuits:

  1. Either we have:
  $$a_{0,0} + \sum_{\ell\in[3]} \gamma_\ell \cdot a_{\ell,0} = a_{0,1} + \sum_{\ell\in[3]} \gamma_\ell \cdot a_{\ell,1},$$
  i.e., equivalently, we have:
  $$\prod_{\ell\in[0,3]} g_\ell^{a_{\ell,0}} = \prod_{\ell\in[0,3]} g_\ell^{a_{\ell,1}}.$$
  2. Or $y \in \mathcal{L}$.

  Formally, π_1 is a verifying NIZK proof under crs_1 of the **OR** relation R_1 defined previously in Appendix 5.

Finally, an encoding $(\mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, \pi_0, \pi_1)$ is said to be "valid" if both of the following hold simultaneously:

$$\mathsf{NIZK.Verify}(\mathsf{crs}_0, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, \{z_{t,j,b}\}_{t\in[|T|],j\in[n],b\in\{0,1\}}), \pi_0) = 1,$$

and

$$\mathsf{NIZK.Verify}(\mathsf{crs}_1, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0, \mathsf{ct}_1, \mathbf{i}, y), \pi_1) = 1.$$

8.4 Operations Over Encodings

Addition, inversion, multiplication, extraction and zero-testing of encodings in the modified construction follow exactly the same procedure as in our original MMap construction in Appendix 5. The circuits for addition, inversion, multiplication, and extraction are exactly as described in Figures 2, 3, 5 and 6, respectively, except that checking the relation $\bar{\mathsf{R}}_1$ in each circuit now uses the matrix of elements $\{z_{t,j}\}_{t\in[|T|],j\in[n]}$. Zero-testing again follows trivially from extraction.

8.5 Proof of Joint-SXDH Hardness

In this section, we prove that solving joint-SXDH is hard over the modified asymmetric MMap construction if solving DDH is hard over the group $\mathbb{G}$. More specifically we state the following theorem:

Theorem 8.1. *The joint-SXDH assumption holds over the modified asymmetric MMap construction provided that all of the following assumptions hold:*

- *The piO scheme is indistinguishability-secure against X-IND samplers.*
- *Membership in $\mathcal{L}$ is computationally hard to decide,*
- *Members of $\mathcal{L}$ have unique witnesses for membership.*
- *The dual-mode NIZK proof system satisfies CRS indistinguishability, perfect extractability in the binding mode, and perfect witness-indistinguishability in the hiding mode.*
- *The DDH assumption holds over the group $\mathbb{G}$.*

We outline the hybrids that are used in the proof. The hybrids are classified into two broad categories - outer hybrids and inner hybrids. We outline the outer hybrids - the inner hybrids are essentially identical to those in the proof of SXDH in Appendix 6.

Outer hybrid 0 corresponds to the game where the challenger provides the adversary with encodings of uniformly random elements in the level-sets that are chosen by the adversary. The final outer hybrid corresponds to the game where the challenger provides the adversary with a valid joint-SXDH instance, i.e., encodings of elements sampled according to the SXDH distribution, in each level-set chosen by the adversary. Before we describe the outer hybrids, we describe a few conditions that remain invariant throughout the outer hybrids:

- The NIZK CRS strings crs_0 and crs_1 are in binding mode, as in the real MMap scheme, in all the outer hybrids.
- The element y in the public parameter is a non-member for the language $\mathcal{L}$, as in the real MMap scheme, in all the outer hybrids.
- The challenge encodings provided to the adversary are *consistent* with respect to extraction (as formalized by the relation R_1 described earlier) in all the outer hybrids.

Table 22 provides an overview of the outer hybrids, which we now describe in details.

Outer Hybrid 0. In this hybrid, the MMap is set up exactly as in the real scheme described earlier. Let (g_0, g_1, g_2, g_3) be the tuple of group elements hardwired into each of the MMap circuits, such that $g_\ell = g_0^{\gamma_\ell}$ for $\ell \in \{1,2,3\}$. Let μ be a uniformly sampled element in $\mathbb{Z}_q$. The joint-SXDH adversary is provided with encodings of the tuple of elements:

$$(\alpha_0, \alpha_1, \alpha_2, \alpha_3) = (\mu, \mu\cdot\gamma_1, \mu\cdot\gamma_2, \mu\cdot\gamma_3),$$

where the encodings are generated in normal form (formalized by relation R_0 described earlier) corresponding to each level-set $\mathbf{i}_t$ for $t \in [|T|]$ chosen by the SXDH adversary. Each encoding also contains a NIZK proof π_0 for normal representation under the binding crs_0, and a NIZK proof π_1 for consistency with respect to extraction under the binding crs_1.

Outer Hybrid	MMap Circuits	$\{z_{t,j,b}\}$	(g_0, g_1, g_2, g_3)	Challenge Encodings			
				Joint-SXDH/Random	Representation	Witness for π_0	Witness for π_1
0	C_{op}	$\{z_{t,j,b} \notin \mathcal{L}\}_{t\in\|T\|, j\in[n], b\in\{0,1\}}$	Random	Random	Normal	$(\mathsf{sk}_0, \mathsf{sk}_1)$	$(\mathsf{sk}_0, \mathsf{sk}_1)$
1	$\widehat{\mathsf{C}}_{\mathsf{op}}$	$\{z_{t,j,b} \in \mathcal{L}\}_{t\in\|T\|, j\in[n], b\in\{0,1\}}$	Random	Random	Normal	$(t, \mathcal{S}_{\mathbf{i}_t}, \{\mathsf{wit}_{\{z_{t,j,0}\}}\}_{j\in\mathcal{S}_{\mathbf{i}_t}})$	$(\mathsf{sk}_0, \mathsf{sk}_1)$
2	$\widehat{\mathsf{C}}_{\mathsf{op}}$	$\{z_{t,j,b} \in \mathcal{L}\}_{t\in\|T\|, j\in[n], b\in\{0,1\}}$	Random	Random	Partially oblique	$(t, \mathcal{S}_{\mathbf{i}_t}, \{\mathsf{wit}_{\{z_{t,j,0}\}}\}_{j\in\mathcal{S}_{\mathbf{i}_t}})$	$(\mathsf{sk}_0, \mathsf{sk}_1)$
3	$\widehat{\mathsf{C}}_{\mathsf{op}}$	$\{z_{t,j,b} \in \mathcal{L}\}_{t\in\|T\|, j\in[n], b\in\{0,1\}}$	Random	Random	Oblique	$(t, \mathcal{S}_{\mathbf{i}_t}, \{\mathsf{wit}_{\{z_{t,j,0}\}}\}_{j\in\mathcal{S}_{\mathbf{i}_t}})$	$(\mathsf{sk}_0, \mathsf{sk}_1)$
4	$\widehat{\widehat{\mathsf{C}}}_{\mathsf{op}}$	$\{\{z_{t,j,0} \in \mathcal{L}\}_{j\in\mathcal{S}_{\mathbf{i}_t}}\}_{t\in\|T\|}$	Random	Random	Oblique	$(t, \mathcal{S}_{\mathbf{i}_t}, \{\mathsf{wit}_{\{z_{t,j,0}\}}\}_{j\in\mathcal{S}_{\mathbf{i}_t}})$	$(\mathsf{sk}_0, \mathsf{sk}_1)$
5	$\widehat{\widehat{\widehat{\mathsf{C}}}}_{\mathsf{op}}$	$\{\{z_{t,j,0} \in \mathcal{L}\}_{j\in\mathcal{S}_{\mathbf{i}_t}}\}_{t\in\|T\|}$	DDH	Joint-SXDH	Oblique	$(t, \mathcal{S}_{\mathbf{i}_t}, \{\mathsf{wit}_{\{z_{t,j,0}\}}\}_{j\in\mathcal{S}_{\mathbf{i}_t}})$	$(\mathsf{sk}_0, \mathsf{sk}_1)$
6	$\widehat{\mathsf{C}}_{\mathsf{op}}$	$\{z_{t,j,b} \in \mathcal{L}\}_{t\in\|T\|, j\in[n], b\in\{0,1\}}$	Random	Random	Oblique	$(t, \mathcal{S}_{\mathbf{i}_t}, \{\mathsf{wit}_{\{z_{t,j,0}\}}\}_{j\in\mathcal{S}_{\mathbf{i}_t}})$	$(\mathsf{sk}_0, \mathsf{sk}_1)$
7	$\widehat{\mathsf{C}}_{\mathsf{op}}$	$\{z_{t,j,b} \in \mathcal{L}\}_{t\in\|T\|, j\in[n], b\in\{0,1\}}$	DDH	Joint-SXDH	Partially oblique	$(t, \mathcal{S}_{\mathbf{i}_t}, \{\mathsf{wit}_{\{z_{t,j,0}\}}\}_{j\in\mathcal{S}_{\mathbf{i}_t}})$	$(\mathsf{sk}_0, \mathsf{sk}_1)$
8	$\widehat{\mathsf{C}}_{\mathsf{op}}$	$\{z_{t,j,b} \in \mathcal{L}\}_{t\in\|T\|, j\in[n], b\in\{0,1\}}$	DDH	Joint-SXDH	Normal	$(t, \mathcal{S}_{\mathbf{i}_t}, \{\mathsf{wit}_{\{z_{t,j,0}\}}\}_{j\in\mathcal{S}_{\mathbf{i}_t}}$	$(\mathsf{sk}_0, \mathsf{sk}_1)$
9	C_{op}	$\{z_{t,j,b} \notin \mathcal{L}\}_{t\in\|T\|, j\in[n], b\in\{0,1\}}$	DDH	Joint-SXDH	Normal	$(\mathsf{sk}_0, \mathsf{sk}_1)$	$(\mathsf{sk}_0, \mathsf{sk}_1)$
10	C_{op}	$\{z_{t,j,b} \notin \mathcal{L}\}_{t\in\|T\|, j\in[n], b\in\{0,1\}}$	Random	Joint-SXDH	Normal	$(\mathsf{sk}_0, \mathsf{sk}_1)$	$(\mathsf{sk}_0, \mathsf{sk}_1)$

Table 22: Overview of the outer hybrids in the proof of joint-SXDH. Changes between subsequent hybrids are highlighted in red. Throughout, crs_0 and crs_1 are binding, $y \notin \mathcal{L}$, and the challenge encodings are consistent with respect to extraction. We use the shorthands C_{op} and $\widehat{\mathsf{C}}_{\mathsf{op}}$ for the tuples $(\mathsf{C}_{\mathsf{Add}}, \mathsf{C}_{\mathsf{Inv}}, \mathsf{C}_{\mathsf{Mult}}, \mathsf{C}_{\mathsf{ext}})$ and $(\widehat{\mathsf{C}}_{\mathsf{Add}}, \widehat{\mathsf{C}}_{\mathsf{Inv}}, \widehat{\mathsf{C}}_{\mathsf{Mult}}, \widehat{\mathsf{C}}_{\mathsf{ext}})$, respectively, where the second set of circuits are as described earlier in the proof of SXDH. Note that for each $t \in [|T|]$, $\mathcal{S}_{\mathbf{i}_t} \in \mathcal{P}([n])$ is the set of all indices $j \in [n]$ such that $\mathbf{i}_{t,j} = 0$.

Outer Hybrid 1. In this hybrid, we make the following alterations to the manner in which the MMap is generated:

1. For each $t \in [|T|]$, $j \in [n]$ and $b \in \{0,1\}$, switch $z_{t,j,b}$ in the public parameters from a non-member to a member for $\mathcal{L}$, i.e., we now sample $z_{t,j,b} \leftarrow \mathcal{L}$, along with a (unique) membership-witness $\mathsf{wit}_{z_{t,j,b}}$.

2. We switch the circuits for addition, inversion, multiplication and extraction as follows:

$$\mathsf{C}_{\mathsf{Add}} \mapsto \widehat{\mathsf{C}}_{\mathsf{Add}} \quad , \quad \mathsf{C}_{\mathsf{Inv}} \mapsto \widehat{\mathsf{C}}_{\mathsf{Inv}},$$
$$\mathsf{C}_{\mathsf{Mult}} \mapsto \widehat{\mathsf{C}}_{\mathsf{Mult}} \quad , \quad \mathsf{C}_{\mathsf{ext}} \mapsto \widehat{\mathsf{C}}_{\mathsf{ext}},$$

The switched circuits are essentially as already described in Figures 7, 8, 9, and 10, respectively, in Appendix 6. At a high level, in each of these switched circuits, we hardwire the witness $\mathsf{wit}_{z_{t,j,b}}$ for the membership of each $z_{t,j,b}$ in $\mathcal{L}$ and avoid using the first extraction trapdoor $\mathsf{t}_{\mathsf{ext},0}$. In other words, we "allow" valid encodings in any level-set $\mathbf{i}'$ to use the oblique representation.

Additionally, we make the following change to the manner in which the challenge encodings are generated: for a challenge-encoding at level $\mathbf{i}_t$, the NIZK proof π_0 under the binding crs_0 now proves language-membership of each $z_{t,j,0}$ such that $\mathbf{i}_{t,j} = 0$, as opposed to proving that the encoding is in the normal representation.

Outer Hybrid 2. This hybrid is identical to the outer hybrid 1, except that the challenge joint-SXDH encodings are no longer in normal form. In particular, the first FHE ciphertext in each encoding now encrypts an oblique representation of the underlying plaintext element. The NIZK proofs continue to be generated as in outer hybrid 1.

Outer Hybrid 3. This hybrid is identical to the outer hybrid 2, except that the challenge joint-SXDH encodings are now entirely in oblique form. In particular, the second FHE ciphertext in each encoding now also encrypts an oblique representation of the underlying plaintext element. The NIZK proofs continue to be generated as in outer hybrid 2.

Outer Hybrid 4. In this hybrid, we make the following alterations to the manner in which the MMap is generated:

1. For each $t \in [|T|]$, let $\mathbf{i}_t$ be the challenge level-set chosen the joint-SXDH adversary. For each $j \in [n]$ such that $\mathbf{i}_{t,j} = 0$, sample $z_{t,j,0}$ in the public parameter as a member for $\mathcal{L}$, and sample all other elements in the public parameter as non-members.

2. We switch the circuits for addition, inversion, multiplication and extraction as follows:

$$\widehat{\mathsf{C}}_{\mathsf{Add}} \mapsto \widehat{\widehat{\mathsf{C}}}_{\mathsf{Add}} \quad , \quad \widehat{\mathsf{C}}_{\mathsf{Inv}} \mapsto \widehat{\widehat{\mathsf{C}}}_{\mathsf{Inv}} \quad , \quad \widehat{\mathsf{C}}_{\mathsf{Mult}} \mapsto \widehat{\widehat{\mathsf{C}}}_{\mathsf{Mult}}.$$

The switched circuits are essentially as already described in Figures 11, 12, and 14, respectively, in Appendix 6. At a high level, we "only allow" valid encodings in a level-set $\mathbf{i}'$ that subsumes at least one challenge level-set to use the oblique representation.

Outer Hybrid 5. This hybrid is identical to outer hybrid 4 except that we switch the tuple (g_0, g_1, g_2, g_3) hardwired into the MMap circuits $\widehat{\widehat{\mathsf{C}}}_{\mathsf{Add}}$, $\widehat{\widehat{\mathsf{C}}}_{\mathsf{Inv}}$, $\widehat{\widehat{\mathsf{C}}}_{\mathsf{Mult}}$ and $\widehat{\mathsf{C}}_{\mathsf{ext}}$ from a uniformly random tuple of group elements to a uniformly random DDH tuple, albeit with the same base element g_0. More formally, we uniformly sample $\gamma_1, \gamma_2 \leftarrow \mathbb{Z}_q$, and set

$$(g_1, g_2, g_3) = (g_0^{\gamma_1}, g_0^{\gamma_2}, g_0^{\gamma_1 \cdot \gamma_2}).$$

Outer Hybrid 6. In this hybrid, we make the following alterations to the manner in which the MMap is generated:

1. For each $t \in [|T|]$, $j \in [n]$ and $b \in \{0, 1\}$, sample $z_{t,j,b}$ in the public parameters as a member for $\mathcal{L}$, i.e., we now sample each $z_{t,j,b} \leftarrow \mathcal{L}$, along with a (unique) membership-witness $\mathsf{wit}_{z_{t,j,b}}$.

2. We switch back the circuits for addition, inversion, multiplication and extraction as follows:

$$\widehat{\widehat{\mathsf{C}}}_{\mathsf{Add}} \mapsto \widehat{\mathsf{C}}_{\mathsf{Add}} \quad , \quad \widehat{\widehat{\mathsf{C}}}_{\mathsf{Inv}} \mapsto \widehat{\mathsf{C}}_{\mathsf{Inv}} \quad , \quad \widehat{\widehat{\mathsf{C}}}_{\mathsf{Mult}} \mapsto \widehat{\mathsf{C}}_{\mathsf{Mult}}.$$

At a high level, we again "allow" valid encodings in any level-set $\mathbf{i}'$ to use the oblique representation.

Outer Hybrid 7. This hybrid is identical to the outer hybrid 6, except that the challenge joint-SXDH encodings are now switched back to a partially oblique form. In particular, the second FHE ciphertext in each encoding now encrypts a normal representation of the underlying plaintext element. The NIZK proofs continue to be generated as in outer hybrid 6.

Outer Hybrid 8. This hybrid is identical to the outer hybrid 7, except that the challenge joint-SXDH encodings are now switched back entirely to the normal form. In particular, the first FHE ciphertext in each encoding now also encrypts a normal representation of the underlying plaintext element. The NIZK proofs continue to be generated as in outer hybrid 7.

Outer Hybrid 9. This hybrid is identical to outer hybrid 8, except that we make the following alterations to the manner in which the MMap is set up:

1. For each $t \in [|T|]$, $j \in [n]$ and $b \in \{0,1\}$, we switch back $z_{t,j,b}$ in the public parameters from a uniform member to a uniform non-member for $\mathcal{L}$, as in the real scheme.

2. We switch back the circuits for addition, inversion, multiplication and extraction as follows:

$$\begin{aligned} \widehat{\mathsf{C}}_{\mathsf{Add}} \mapsto \mathsf{C}_{\mathsf{Add}} \quad &, \quad \widehat{\mathsf{C}}_{\mathsf{Inv}} \mapsto \mathsf{C}_{\mathsf{Inv}}, \\ \widehat{\mathsf{C}}_{\mathsf{Mult}} \mapsto \mathsf{C}_{\mathsf{Mult}} \quad &, \quad \widehat{\mathsf{C}}_{\mathsf{ext}} \mapsto \mathsf{C}_{\mathsf{ext}}, \end{aligned}$$

   In particular, the circuits are now exactly as in the real MMap scheme, with the exception that the tuple (g_0, g_1, g_2, g_3) hardwired inside these circuits continues to be a DDH tuple.

Additionally, we make the following change to the manner in which the challenge encodings are generated: for each challenge encoding, the NIZK proof π_0 under the binding crs_0 now proves that the encoding is in the normal representation using the witness $(\mathsf{sk}_0, \mathsf{sk}_1)$.

Outer Hybrid 10. This hybrid is identical to outer hybrid 7 except that we switch the tuple of group elements (g_0, g_1, g_2, g_3) hardwired into the MMap circuits $\mathsf{C}_{\mathsf{Add}}$, $\mathsf{C}_{\mathsf{Inv}}$, $\mathsf{C}_{\mathsf{Mult}}$ and $\mathsf{C}_{\mathsf{ext}}$ from a uniform DDH tuple to a uniformly random tuple, albeit with the same base element g_0. In other words, the MMap circuits are now exactly as in the real scheme.

Indistinguishability of Outer Hybrids. The proof of indistinguishability of the outer hybrids are very similar to the proof of indistinguishability of the outer hybrids in the proof of SXDH (Appendix 6). Hence we do not detail them.

9 A Symmetric MMap Construction

In this section, we show a how to construct a symmetric MMap given same set of cryptoprimitives as in the asymmetric MMap construction:

- A probabilistic-iO scheme piO $= (\mathsf{piO.Obf}, \mathsf{piOEval})$.
- A fully-homomorphic encryption scheme

$$\text{FHE} = (\mathsf{FHE.Gen}, \mathsf{FHE.Enc}, \mathsf{FHE.Dec}, \mathsf{FHE.Eval}),$$

  such that the message space is $\mathbb{Z}_q$ for some prime $q = \mathrm{poly}(\lambda)$ (λ being the security parameter).

- A dual mode NIZK argument system NIZK $= (\mathsf{NIZK.Setup}, \mathsf{NIZK.Prove}, \mathsf{NIZK.Verify})$ that is:
  - perfectly sound and extractable in the binding mode, and
  - perfectly witness indistinguishable and perfectly zero-knowledge in the hiding mode.

- 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 $y \in \mathcal{L}$.

- A pairing-free group $\mathbb{G}$ of prime order q.

In Appendix 10 we show that $(n+1)$-EDDH is hard over our proposed MMap construction if CP-EDDH is hard over the group $\mathbb{G}$.

Differences with Asymmetric MMap Construction. At a high level, our symmetric MMap construction is very similar to the asymmetric MMap construction presented in Appendix 5, except for the following key differences that we discuss informally here.

"All-or-Nothing" Approach to Representation. In the asymmetric MMap construction and the corresponding proof of SXDH (and k-EDDH), we used an OR-proof technique to restrict the "validity" of encodings in (partially) oblique representation to particular level-sets. In other words, depending on the public parameters for the MMap scheme, either all valid encodings were in the normal representation, or valid encodings in certain level-sets were allowed to be in the oblique representation. In the symmetric MMap construction, where no such level-sets exist, such restrictions cannot be imposed. Hence, we opt for an "all-or-nothing" approach - depending on the public parameters for the MMap scheme, either all valid encodings must be in the normal representation, or all valid encodings are allowed to be in the oblique representation. This also means that it now suffices to have a single element $z \in \mathcal{X}$ in the public parameter in order to "switch-on" or "switch-off" this OR branch, as opposed to n elements $z_1, \ldots, z_n$ in the asymmetric construction.

"Well-Formedness". The aforementioned change, however, presents certain challenges when multiplying encodings. In the asymmetric construction, we ensured that any two "allowed" level-sets for oblique representation must be incompatible with respect to multiplication. Since we cannot enforce such a restriction in the symmetric setting, we opt for a different restriction, called "well-formedness", that must be satisfied irrespective of whether the encoding is in normal or oblique representation. Any encoding in normal representation is well-formed by default. On the other hand, any pair of oblique encodings can be multiplied provided that they are well-formed. The formal definition of well-formedness is presented subsequently.

9.1 Setup

The setup algorithm for our MMap construction takes as input the security parameter 1^λ and a second parameter 1^n for the degree of multilinearity. It samples two key-pairs for the FHE scheme as:

$$(\mathsf{pk}_0, \mathsf{sk}_0), (\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathsf{FHE.Gen}(1^\lambda).$$

Unlike the asymmetric construction, the setup algorithm for our symmetric MMap construction uniformly samples $\gamma, \delta \leftarrow \mathbb{Z}_q$ and $g \leftarrow \mathbb{G}$, and creates a tuple of $n(n+1)/2$ group elements in $\mathbb{G}$ as:

$$g_{\ell,\ell'} = g^{\gamma^{\ell} \cdot \delta^{\ell'}} \text{ for each } \ell, \ell' \in [0, n] \text{ such that } \ell + \ell' \leq n.$$

As explained subsequently, this change is motivated by the nature of the hardness assumption we wish to prove over our symmetric MMap construction.

The setup algorithm also samples a pair of binding NIZK CRS strings (along with the corresponding extraction trapdoors) as:

$$(\mathsf{crs}_0, \mathsf{t}_{\mathsf{ext},0}), (\mathsf{crs}_1, \mathsf{t}_{\mathsf{ext},1}) \leftarrow \mathsf{NIZK.Setup}(1^\lambda, \mathsf{binding}).$$

The languages for which statements are proven under these CRS strings are described subsequently in Section 9.2. Next the setup algorithm uniformly samples a pair of non-member elements as:

$$y, z \leftarrow \mathcal{X} \setminus \mathcal{L}.$$

Finally, the setup algorithm computes and outputs four probabilistically indistinguishability-obfuscated circuits of the form

$$\bar{\mathsf{C}}_{\mathsf{Add}} = \mathsf{piO.Obf}(\mathsf{C}_{\mathsf{Add}}) \quad , \quad \bar{\mathsf{C}}_{\mathsf{Inv}} = \mathsf{piO.Obf}(\mathsf{C}_{\mathsf{Inv}}),$$
$$\bar{\mathsf{C}}_{\mathsf{Mult}} = \mathsf{piO.Obf}(\mathsf{C}_{\mathsf{Mult}}) \quad , \quad \bar{\mathsf{C}}_{\mathsf{ext}} = \mathsf{piO.Obf}(\mathsf{C}_{\mathsf{ext}})$$

where $\mathsf{C}_{\mathsf{Add}}$, $\mathsf{C}_{\mathsf{Inv}}$ $\mathsf{C}_{\mathsf{Mult}}$, and $\mathsf{C}_{\mathsf{ext}}$ are the circuits for adding, inverting, multiplying, and extracting from encodings at any given "non-zero" level. Note that while the names for these circuits are overloaded from the asymmetric MMap construction for ease of representation, their descriptions are not exactly identical to their counterparts described previously. We describe these circuits in details subsequently. However, as before, these circuits also embed certain elements that we want to keep secret:

- The FHE secret keys sk_0 and sk_1.
- The NIZK extraction trapdoors $\mathsf{t}_{\mathsf{ext},0}$ and $\mathsf{t}_{\mathsf{ext},1}$.
- The tuple of group elements $\{g_{\ell,\ell'}\}_{\ell+\ell'\in[1,n]}$ (only $g_{0,0}$ is made publicly available).

This is why these circuits are not made public as is. Instead, the setup algorithm only makes available piO-obfuscated versions of these circuit. To summarize, the public parameter pp for our symmetric MMap construction is as follows:

$$\mathsf{pp} = \left(\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{crs}_0, \mathsf{crs}_1, g_{0,0}, y, z, \bar{\mathsf{C}}_{\mathsf{Add}}, \bar{\mathsf{C}}_{\mathsf{Inv}}, \bar{\mathsf{C}}_{\mathsf{Mult}}, \bar{\mathsf{C}}_{\mathsf{ext}}\right).$$

9.2 Encodings

We describe the procedure of encoding a plaintext element at any level i such that $i \in [0, n]$. In our construction, level-0 encodings are treated slightly different from encodings at other "non-zero" levels, and are equipped with their own set of algorithms for encoding, manipulation, extraction and zero-testing. We informally mention these for the sake of completeness.

Level-Zero Encodings. We set the level-0 encoding of a plaintext element $a \in \mathbb{Z}_q$ to be a itself. Adding/multiplying two level-0 encodings (equivalently, additively inverting a level-0 encoding) is simply done via addition/multiplication (equivalently, additive inversion) in $\mathbb{Z}_q$.

Multiplying a level-0 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.

Extracting a level-0 encoding $a \in \mathbb{Z}_q$ outputs $g_{0,0}^a$, where $g_{0,0} \in \mathbb{G}$ is the publicly available group element sampled at setup that is also embedded inside each MMap circuit. As will be clear later, this is consistent with the extraction algorithm for any other level i. Zero-testing follows trivially from the extraction algorithm.

Level-i Encodings. We now describe the procedure of encoding a plaintext element at any "non-zero" encoding level i such that $i \in [n]$. An encoding of a plaintext element $a \in \mathbb{Z}_q$ at level-vector i is a tuple of the form $(\mathsf{ct}_0, \mathsf{ct}_1, i, \pi_0, \pi_1)$, where:

- ct_0 and ct_1 are FHE encryptions of a tuple of the form:

  $$(\{a_{\ell,\ell',0}\}_{\ell+\ell'\in[0,n]}, i) \quad , \quad (\{a_{\ell,\ell',1}\}_{\ell+\ell'\in[0,n]}, i).$$

  under the public key-secret key pairs $(\mathsf{pk}_0, \mathsf{sk}_0)$ and $(\mathsf{pk}_1, \mathsf{sk}_1)$ respectively.

  For $b \in \{0, 1\}$, the tuple $(\{a_{\ell,\ell',b}, i)$ is said to be in "normal form" if

  $$\{a_{\ell,\ell',b} = 0\}_{(\ell,\ell')\neq(0,0)}.$$

  Otherwise, it is said to be in "oblique-form". As before, depending on the forms of the tuples underlying the FHE ciphertexts, we classify an encoding into one of three representations:

  – **Normal representation:** Both tuples are in normal form.

  – **Partially oblique representation:** Exactly one of the tuples is in normal form, while the other is in oblique form. Unless otherwise specified, we assume that the second tuple is in normal form.

  – **Oblique representation:** Both tuples are in oblique form.

We also introduce the concept of a "well-formed encoding". An encoding at level $i \in [n]$ is said to be "well-formed" if the tuples underlying the FHE ciphertexts satisfy the following condition:

$$a_{\ell,\ell',b} = 0 \text{ for each } \ell, \ell' \in [0,n] \text{ such that } \ell + \ell' > i.$$

Also recall that the setup algorithm privately samples a tuple of group elements $(\{g_{\ell,\ell'}\})$ and hardwires these into the MMap circuits that are described subsequently. We say that an encoding is "consistent" if the tuples underlying the FHE ciphertexts satisfy the following condition:

$$\prod_{\ell+\ell' \in [0,n]} (g_{\ell,\ell'})^{a_{\ell,\ell',0}} = \prod_{\ell+\ell' \in [0,n]} (g_{\ell,\ell'})^{a_{\ell,\ell',1}}.$$

Looking ahead, this will be used explicitly in the extraction algorithm of our construction.

- π_0 is a verifying NIZK proof of "normal representation". Informally, it proves under crs_0 that one of the following statements must be true: either the encoding is in normal representation or $z \in \mathcal{L}$. Formally, π_0 is a verifying NIZK proof under crs_0 of the **OR** relation $\bar{\mathsf{R}}_0$ defined below ($\mathcal{K}_{\text{FHE}}$ being the set of all valid key pairs under the FHE scheme and $\mathsf{R}_{\mathcal{L}}$ being as defined earlier):

Relation R_0:

$\bar{\mathsf{R}}_0((\mathsf{ct}_0, \mathsf{ct}_1, i, \mathsf{pk}_0, \mathsf{pk}_1), \mathsf{wit}) = 1$ if and only if:

- **Either** $\mathsf{wit} = (\mathsf{sk}_0, \mathsf{sk}_1)$ *and* $(\mathsf{pk}_0, \mathsf{sk}_0), (\mathsf{pk}_1, \mathsf{sk}_1) \in \mathcal{K}_{\text{FHE}}$ *and*

$$\mathsf{FHE.Dec}(\mathsf{sk}_0, \mathsf{ct}_0) = \mathsf{FHE.Dec}(\mathsf{sk}_1, \mathsf{ct}_1) = (a, 0, 0, \ldots, 0, i) \text{ for some } a \in \mathbb{Z}_q.$$

- **Or** $\mathsf{wit} = (a, \mathsf{r}_0, \mathsf{r}_1)$ for some $a \in \mathbb{Z}_q$ *and* for each $b \in \{0,1\}$, we have:

$$\mathsf{FHE.Enc}(\mathsf{pk}_b, (a, 0, 0, \ldots, 0, i); \mathsf{r}_b) = \mathsf{ct}_b.$$

OR Relation $\bar{\mathsf{R}}_0$:

$$\bar{\mathsf{R}}_0((\mathsf{ct}_0, \mathsf{ct}_1, i, \mathsf{pk}_0, \mathsf{pk}_1, z), \mathsf{wit}) = \mathsf{R}_0((\mathsf{ct}_0, \mathsf{ct}_1, i, \mathsf{pk}_0, \mathsf{pk}_1), \mathsf{wit}) \vee \mathsf{R}_{\mathcal{L}}(z, \mathsf{wit}).$$

- π_1 is a verifying NIZK proof of "well-formedness" *and* "consistency" that holds irrespective of whether the encoding is in normal representation or (partially) oblique representation. Informally, it proves under crs_1 that one of the following statements must be true: either the encoding is well-formed and consistent, or $y \in \mathcal{L}$. Formally, π_1 is a verifying NIZK proof under crs_1 of the **OR** relation $\bar{\mathsf{R}}_1$ defined below ($\mathsf{R}_{\mathcal{L}}$ is as defined earlier):

Relation R_1:

$\mathsf{R}_1((\mathsf{ct}_0, \mathsf{ct}_1, i, \mathsf{pk}_0, \mathsf{pk}_1), \mathsf{wit}) = 1$ if and only if:

- **Either** $\mathsf{wit} = (\mathsf{sk}_0, \mathsf{sk}_1)$ *and* $(\mathsf{pk}_0, \mathsf{sk}_0), (\mathsf{pk}_1, \mathsf{sk}_1) \in \mathcal{K}_{\mathrm{FHE}}$ *and* for $b \in \{0, 1\}$, we have

$$\mathsf{FHE.Dec}(\mathsf{sk}_b, \mathsf{ct}_b) = (\{a_{\ell,\ell',b}\}_{\ell+\ell' \in [0,n]}, i),$$

such that for we have

$$a_{\ell,\ell',b} = 0 \text{ for each } \ell, \ell' \in [0, n] \text{ such that } \ell + \ell' \geq i,$$

and we have

$$\prod_{\ell+\ell' \in [0,n]} (g_{\ell,\ell'})^{a_{\ell,\ell',0}} = \prod_{\ell+\ell' \in [0,n]} g_{\ell,\ell',1}^{a_{\ell,\ell',1}} = g_{0,0}^{a}.$$

- **Or** $\mathsf{wit} = (\{a_{\ell,\ell',0}, a_{\ell,\ell',1}\}_{\ell+\ell' \in [0,n]}, \mathsf{r}_0, \mathsf{r}_1)$ *and* for each $b \in \{0, 1\}$, we have

$$\mathsf{FHE.Enc}(\mathsf{pk}_b, (\{a_{\ell,\ell',b}\}_{\ell+\ell' \in [0,n]}, i); \mathsf{r}_b) = \mathsf{ct}_b,$$

and we have

$$a_{\ell,\ell',b} = 0 \text{ for each } \ell, \ell' \in [0, n] \text{ such that } \ell + \ell' \geq i,$$

and we have

$$\prod_{\ell+\ell' \in [0,n]} (g_{\ell,\ell'})^{a_{\ell,\ell',0}} = \prod_{\ell+\ell' \in [0,n]} (g_{\ell,\ell'})^{a_{\ell,\ell',1}},$$

OR Relation $\bar{\mathsf{R}}_1$:

$$\bar{\mathsf{R}}_1((\mathsf{ct}_0, \mathsf{ct}_1, i, \mathsf{pk}_0, \mathsf{pk}_1, y), \mathsf{wit}) = \mathsf{R}_1((\mathsf{ct}_0, \mathsf{ct}_1, i, \mathsf{pk}_0, \mathsf{pk}_1), \mathsf{wit}) \vee \mathsf{R}_{\mathcal{L}}(y, \mathsf{wit}).$$

An encoding $(\mathsf{ct}_0, \mathsf{ct}_1, i, \pi_0, \pi_1)$ is said to be "valid" if both of the following hold simultaneously:

$$\mathsf{NIZK.Verify}(\mathsf{crs}_0, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0, \mathsf{ct}_1, i, z), \pi_0) = 1,$$

and

$$\mathsf{NIZK.Verify}(\mathsf{crs}_1, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0, \mathsf{ct}_1, i, y), \pi_1) = 1.$$

Note that when $z \notin \mathcal{L}$ and $y \notin \mathcal{L}$, any valid encoding must be in the normal representation, and hence must also be well-formed and consistent. So having the additional proof π_1 might appear redundant. However, looking ahead, during certain hybrids in the proof of security, we will relax one or more of the language non-membership conditions to allow encodings corresponding to certain designated level sets to be in oblique representation. In such a case, the proof π_1 would allow us to enforce well-formedness and consistency irrespective of whether the encoding is in the normal representation or in the oblique representation.

9.3 Addition and Inversion of Encodings

We now describe the procedure for adding two encodings, and for (additive) inversion of an encoding. Suppose we have two encodings at the same level i of the form:

$$(\mathsf{ct}_{0,1}, \mathsf{ct}_{1,1}, i, \pi_{0,1}, \pi_{1,1}) \quad , \quad (\mathsf{ct}_{0,2}, \mathsf{ct}_{1,2}, i, \pi_{0,2}, \pi_{1,2}).$$

Figure 20 details the operation of the encoding-addition circuit $\mathsf{C}_{\mathsf{Add}}$. Note that it embeds multiple secrets, including the FHE secret keys sk_0 and sk_1, as well as the extraction trapdoors $\mathsf{t}_{\mathsf{ext},0}$ and $\mathsf{t}_{\mathsf{ext},1}$ 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. The same holds for the encoding-inversion circuit $\mathsf{C}_{\mathsf{Inv}}$, which is described in Figure 21.

At a high level, we add the two input encodings by exploiting the fully-homomorphic nature of the encryption scheme. More concretely, we homomorphically evaluate the circuit $\mathsf{C}_{\mathsf{Add},\mathrm{FHE}}$ (described in Figure 19) on the corresponding ciphertext components of the two input encodings to generate the ciphertext components for the output

$\mathsf{C}_{\mathsf{Add},\mathbf{FHE}}((\{a_{\ell,\ell',1}\}_{\ell+\ell'\in[0,n]}, i_1), (\{a_{\ell,\ell',2}\}_{\ell+\ell'\in[0,n]}, i_2))$**:**

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

$\mathsf{C}_{\mathsf{Inv},\mathbf{FHE}}(\{a_{\ell,\ell'}\}_{\ell+\ell'\in[0,n]}, i)$**:**

1. If $i > n$, output $\perp$.
2. Else, output $\left(\{-a_{\ell,\ell'} \mod q\}_{\ell+\ell'\in[0,n]}, i\right)$.

Figure 19: Circuits $\mathsf{C}_{\mathsf{Add},\mathrm{FHE}}$ and $\mathsf{C}_{\mathsf{Inv},\mathrm{FHE}}$ for the symmetric MMap

encoding. We also generate proofs for normal representation and well-formedness + consistency of the output encoding using the tuple of secret keys $(\mathsf{sk}_0, \mathsf{sk}_1)$ as witness, unless otherwise dictated by the input encodings (in which case we use the extracted witnesses from the proofs in the input encodings to generate the proofs for the output encodings). The approach for inverting an input encoding is very similar, except that we homomorphically evaluate the circuit $\mathsf{C}_{\mathsf{Inv},\mathrm{FHE}}$ (also described in Figure 19) on the ciphertext components of the input encoding.

For technical reasons that are relevant to the proof of security, we check the following in both the addition and inversion circuits:

1. The validity of the proofs π_1 and π_2 that are provided as part of the input encodings (steps 1 and 2).
2. Whether the encodings are in the normal representation as per the relation R_0 described earlier (step 5).
3. Whether the encodings are well-formed and consistent as per the relation R_1 described earlier (step 7).

The checks in steps 5 and 7 of $\mathsf{C}_{\mathsf{Add}}$ and $\mathsf{C}_{\mathsf{Inv}}$ are included for technical reasons that are relevant to the proof of security. In particular, we emphasize the following:

- Checking the "**If**" condition in step 7 of $\mathsf{C}_{\mathsf{Add}}$ and $\mathsf{C}_{\mathsf{Inv}}$ requires the tuple of group elements $\{g_{\ell,\ell'}\}$ sampled at setup, and we implicitly assume that it is embedded in both circuits. We omit explicitly describing it for the sake of brevity.
- When the element z is a non-member for the language $\mathcal{L}$, then under a binding crs_0, the "**If**" condition in step 5 of $\mathsf{C}_{\mathsf{Add}}$ is never satisfied. This follows from the perfect soundness guarantee of the NIZK proof system. However, the condition may be satisfied during some hybrid in the proof of security, when we deliberately switch z to a member of $\mathcal{L}$.
- When the element y is a non-member for the language $\mathcal{L}$, then under a binding crs_1, the "**If**" condition in step 7 of $\mathsf{C}_{\mathsf{Add}}$ is never satisfied. This again follows from the perfect soundness guarantee of the NIZK proof system. However, the condition may be satisfied during some hybrid in the proof of security, when we deliberately switch y to a member of $\mathcal{L}$.

9.4 Multiplication of Encodings

We now describe the procedure for multiplying two encodings at levels i_1 and i_2, respectively such that $i_1 + i_2 \leq n$. Suppose we have two encodings of the form:

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

```
$\mathsf{C}_{\mathsf{Add}}[\{\mathsf{sk}_b, \mathsf{pk}_b, \mathsf{crs}_b, \mathsf{t}_{\mathsf{ext},b}\}_{b\in\{0,1\}}, \{g_{\ell,\ell'}\}]((\mathsf{ct}_{0,1}, \mathsf{ct}_{1,1}, i, \pi_{0,1}, \pi_{1,1}), (\mathsf{ct}_{0,2}, \mathsf{ct}_{1,2}, i, \pi_{0,2}, \pi_{1,2}))$:

1. Output $\perp$ if for any $k \in \{1,2\}$, $\mathsf{NIZK.Verify}(\mathsf{crs}_0, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_{0,k}, \mathsf{ct}_{1,k}, i, z), \pi_{0,k}) = 0$.
2. Output $\perp$ if for any $k \in \{1,2\}$, $\mathsf{NIZK.Verify}(\mathsf{crs}_1, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_{0,k}, \mathsf{ct}_{1,k}, i, y), \pi_{1,k}) = 0$.
3. For $b \in \{0,1\}$, set: $\mathsf{ct}_b^* = \mathsf{FHE.Eval}(\mathsf{pk}_b, \mathsf{ct}_{b,1}, \mathsf{ct}_{b,2}, \mathsf{C}_{\mathsf{Add},\mathrm{FHE}})$.
4. For $b \in \{0,1\}$ and $k \in \{1,2\}$, recover $(\{a_{\ell,\ell',b,k}\}_{\ell+\ell'\in[0,n]}, i) = \mathsf{FHE.Dec}(\mathsf{sk}_b, \mathsf{ct}_{b,k})$.
5. If for some $k \in \{1,2\}$, $\mathsf{R}_0((\mathsf{sk}_0, \mathsf{sk}_1), (\mathsf{ct}_{0,k}, \mathsf{ct}_{1,k}, i, \mathsf{pk}_0, \mathsf{pk}_1)) = 0$, then:
    (a) Extract $\mathsf{wit}_z = \mathsf{NIZK.Ext}(\mathsf{t}_{\mathsf{ext},0}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_{0,1}, \mathsf{ct}_{1,1}, i, z), \pi_{0,1})$.
    (b) If $\mathsf{R}_{\mathcal{L}}(z, \mathsf{wit}_z) = 0$, output $\perp$.
    (c) Else, generate $\pi_0^* \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}_0, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0^*, \mathsf{ct}_1^*, i, z), \mathsf{wit}_z)$.
6. Else, generate $\pi_0^* \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0^*, \mathsf{ct}_1^*, i, z), (\mathsf{sk}_0, \mathsf{sk}_1))$.
7. If for some $k \in \{1,2\}$, we have $\mathsf{R}_1((\mathsf{sk}_0, \mathsf{sk}_1), (\mathsf{ct}_{0,k}, \mathsf{ct}_{1,k}, i, \mathsf{pk}_0, \mathsf{pk}_1)) = 0$, then:
    (a) Extract $\mathsf{wit}_y = \mathsf{NIZK.Ext}(\mathsf{t}_{\mathsf{ext},1}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_{0,1}, \mathsf{ct}_{1,1}, i, y), \pi_{1,1})$.
    (b) If $\mathsf{R}_{\mathcal{L}}(y, \mathsf{wit}_y) = 0$, output $\perp$.
    (c) Else, generate $\pi_1^* \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}_1, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0^*, \mathsf{ct}_1^*, i, y), \mathsf{wit}_y)$.
8. Else, generate $\pi_1^* \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0^*, \mathsf{ct}_1^*, i, y), (\mathsf{sk}_0, \mathsf{sk}_1))$.
9. Output $(\mathsf{ct}_0^*, \mathsf{ct}_1^*, i, \pi_0^*, \pi_1^*)$.
```

Figure 20: Circuit $\mathsf{C}_{\mathsf{Add}}$ for the symmetric MMap

Figure 23 details the operation of the encoding multiplication circuit $\mathsf{C}_{\mathsf{Mult}}$. Note that it again embeds multiple secrets, including the FHE secret keys sk_0 and sk_1, as well as the extraction trapdoors $\mathsf{t}_{\mathsf{ext},0}$ and $\mathsf{t}_{\mathsf{ext},1}$ 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.

```
$\mathsf{C}_{\mathsf{Mult},\mathbf{FHE}}((\{a_{\ell,\ell',1}\}_{\ell+\ell'\in[0,n]}, i_1), (\{a_{\ell,\ell',2}\}_{\ell+\ell'\in[0,n]}, i_2))$:

1. If $i_1 + i_2 > n$, output $\perp$.
2. Else if $a_{\ell,\ell',1} \neq 0$ for some $\ell, \ell'$ such that $\ell + \ell' \geq i_1$, then output $\perp$.
3. Else if $a_{\ell,\ell',2} \neq 0$ for some $\ell, \ell'$ such that $\ell + \ell' \geq i_2$, then output $\perp$.
4. Else, output $(\{a^*_{\ell,\ell'}\}_{(\ell+\ell')\in[0,n]}, (i_1 + i_2))$, where for each $(\ell, \ell')$, we have
   $$a^*_{\ell,\ell'} = \sum_{\substack{\ell_1,\ell_2\in[0,\ell] \text{ s.t. } \ell_1+\ell_2=\ell \\ \ell'_1,\ell'_2\in[0,\ell'] \text{ s.t. } \ell'_1+\ell'_2=\ell'}} a_{\ell_1,\ell'_1,1} \cdot a_{\ell_2,\ell'_2,2}.$$
```

Figure 22: Circuit $\mathsf{C}_{\mathsf{Mult},\mathrm{FHE}}$ for the symmetric MMap

At a high level, we multiply the two input encodings by again exploiting the fully-homomorphic nature of the encryption scheme. More concretely, we homomorphically evaluate the circuit $\mathsf{C}_{\mathsf{Mult},\mathrm{FHE}}$ (described in Figure 22) on the corresponding ciphertext components of the two input encodings to generate the ciphertext components for

```
$\underline{\mathsf{C_{Inv}}[\{\mathsf{sk}_b, \mathsf{pk}_b, \mathsf{crs}_b, \mathsf{t}_{\mathsf{ext},b}\}_{b\in\{0,1\}}, \{g_{\ell,\ell'}\}](\mathsf{ct}_0, \mathsf{ct}_1, i, \pi_0, \pi_1)\text{:}}$

  1. Output $\perp$ if $\mathsf{NIZK.Verify}(\mathsf{crs}_0, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0, \mathsf{ct}_1, i, z), \pi_0) = 0$.
  2. Output $\perp$ if $\mathsf{NIZK.Verify}(\mathsf{crs}_1, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0, \mathsf{ct}_1, i, y), \pi_1) = 0$.
  3. For $b \in \{0,1\}$, set: $\mathsf{ct}^*_b = \mathsf{FHE.Eval}(\mathsf{pk}_b, \mathsf{ct}_b, \mathsf{C}_{\mathsf{Add},\mathrm{FHE}})$.
  4. For $b \in \{0,1\}$, recover $(\{a_{\ell,\ell',b}\}_{\ell+\ell'\in[0,n]}, i) = \mathsf{FHE.Dec}(\mathsf{sk}_b, \mathsf{ct}_b)$.
  5. If $\mathsf{R}_0((\mathsf{sk}_0, \mathsf{sk}_1), (\mathsf{ct}_0, \mathsf{ct}_1, i, \mathsf{pk}_0, \mathsf{pk}_1)) = 0$, then:
     (a) Extract $\mathsf{wit}_z = \mathsf{NIZK.Ext}(\mathsf{t}_{\mathsf{ext},0}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0, \mathsf{ct}_1, i, z), \pi_0)$.
     (b) If $\mathsf{R}_{\mathcal{L}}(z, \mathsf{wit}_z) = 0$, output $\perp$.
     (c) Else, generate $\pi^*_0 \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}_0, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}^*_0, \mathsf{ct}^*_1, i, z), \mathsf{wit}_z)$.
  6. Else, generate $\pi^*_0 \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}^*_0, \mathsf{ct}^*_1, i, z), (\mathsf{sk}_0, \mathsf{sk}_1))$.
  7. If $\mathsf{R}_1((\mathsf{sk}_0, \mathsf{sk}_1), (\mathsf{ct}_0, \mathsf{ct}_1, i, \mathsf{pk}_0, \mathsf{pk}_1)) = 0$, then:
     (a) Extract $\mathsf{wit}_y = \mathsf{NIZK.Ext}(\mathsf{t}_{\mathsf{ext},1}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0, \mathsf{ct}_1, i, y), \pi_1)$.
     (b) If $\mathsf{R}_{\mathcal{L}}(y, \mathsf{wit}_y) = 0$, output $\perp$.
     (c) Else, generate $\pi^*_1 \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}_1, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}^*_0, \mathsf{ct}^*_1, i, y), \mathsf{wit}_y)$.
  8. Else, generate $\pi^*_1 \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}^*_0, \mathsf{ct}^*_1, i, y), (\mathsf{sk}_0, \mathsf{sk}_1))$.
  9. Output $(\mathsf{ct}^*_0, \mathsf{ct}^*_1, i, \pi^*_0, \pi^*_1)$.
```

Figure 21: Circuit $\mathsf{C_{Inv}}$ for the symmetric MMap

the output encoding. Note that the circuit outputs $\perp$ unless both the input encodings are well-formed. However, as described subsequently, we ensure in our MMap construction as well as in the proof of SXDH that any "valid" encoding must be well-formed. Hence, evaluation using the circuit $\mathsf{C}_{\mathsf{Mult},\mathrm{FHE}}$ never outputs $\perp$.

Finally, similar to the addition and inversion circuits, we also generate proofs for normal representation and well-formedness + consistency of the output encoding using the tuple of secret keys $(\mathsf{sk}_0, \mathsf{sk}_1)$ as witness, unless otherwise dictated by the input encodings (in which case we use the extracted witnesses from the proofs in the input encodings to generate the proofs for the output encodings).

Similar to the addition and inversion procedures described previously, the checks in steps 5 and 7 of $\mathsf{C_{Inv}}$ are included for technical reasons that are relevant to the proof of security. In particular, we emphasize the following:

- Checking the "**If**" condition in step 7 of $\mathsf{C_{Mult}}$ requires the tuple of group elements $\{g_{\ell,\ell'}\}$ sampled at setup, and we implicitly assume that it is embedded in the $\mathsf{C_{Mult}}$ circuit. We omit explicitly describing it for the sake of brevity.

- When the element z is a non-member for the language $\mathcal{L}$, then under a binding crs_0, the "**If**" condition in step 5 of $\mathsf{C_{Inv}}$ is never satisfied. This follows from the perfect soundness guarantee of the NIZK proof system, and ensures that no pair of valid input encodings (even adversarially created) can result in the homomorphic evaluation of $\mathsf{C}_{\mathsf{Mult},\mathrm{FHE}}$ outputting $\perp$.

  Looking ahead, in certain hybrids of our proof of SXDH, we do allow the "**If**" condition in step 5 to be satisfiable for encodings corresponding to certain level sets. In these hybrids, we switch exactly z from a non-member to a member for $\mathcal{L}$. Note, however, that in this case, any valid encoding that is allowed to deviate from the normal representation must still be well-formed and consistent.

```
$\mathsf{C}_{\mathsf{Mult}}[\{\mathsf{sk}_b, \mathsf{pk}_b, \mathsf{crs}_b, \mathsf{t}_{\mathsf{ext},b}\}_{b\in\{0,1\}}, \{g_{\ell,\ell'}\}]((\mathsf{ct}_{0,1}, \mathsf{ct}_{1,1}, i_1, \pi_{0,1}, \pi_{1,1}), (\mathsf{ct}_{0,2}, \mathsf{ct}_{1,2}, i_2, \pi_{0,2}, \pi_{1,2}))$:

    1. Output $\perp$ if for any $k \in \{1,2\}$, $\mathsf{NIZK.Verify}(\mathsf{crs}_0, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_{0,k}, \mathsf{ct}_{1,k}, i_k, z), \pi_{0,k}) = 0$.
    2. Output $\perp$ if for any $k \in \{1,2\}$, $\mathsf{NIZK.Verify}(\mathsf{crs}_1, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_{0,k}, \mathsf{ct}_{1,k}, i_k, y), \pi_{1,k}) = 0$.
    3. For $b \in \{0,1\}$, set: $\mathsf{ct}_b^* = \mathsf{FHE.Eval}(\mathsf{pk}_b, \mathsf{ct}_{b,1}, \mathsf{ct}_{b,2}, \mathsf{C}_{\mathsf{Mult,FHE}})$.
    4. For $b \in \{0,1\}$ and $k \in \{1,2\}$, recover $(\{a_{\ell,\ell',b,k}\}_{\ell+\ell'\in[0,n]}, i_k) = \mathsf{FHE.Dec}(\mathsf{sk}_b, \mathsf{ct}_{b,k})$.
    5. If for some $k \in \{1,2\}$, $\mathsf{R}_0((\mathsf{sk}_0, \mathsf{sk}_1), (\mathsf{ct}_{0,k}, \mathsf{ct}_{1,k}, i_k, \mathsf{pk}_0, \mathsf{pk}_1)) = 0$, then:
        (a) Extract $\mathsf{wit}_z = \mathsf{NIZK.Ext}(\mathsf{t}_{\mathsf{ext},0}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_{0,k}, \mathsf{ct}_{1,k}, i_k, z), \pi_{0,k})$.
        (b) If $\mathsf{R}_{\mathcal{L}}(z, \mathsf{wit}_z) = 0$, output $\perp$.
        (c) Else, generate $\pi_0^* \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}_0, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0^*, \mathsf{ct}_1^*, (i_1 + i_2), z), \mathsf{wit}_z)$.
    6. Else, generate $\pi_0^* \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0^*, \mathsf{ct}_1^*, (i_1 + i_2), z), (\mathsf{sk}_0, \mathsf{sk}_1))$.
    7. If for some $k \in \{1,2\}$, we have $\mathsf{R}_1((\mathsf{sk}_0, \mathsf{sk}_1), (\mathsf{ct}_{0,k}, \mathsf{ct}_{1,k}, i_k, \mathsf{pk}_0, \mathsf{pk}_1)) = 0$, then:
        (a) Extract $\mathsf{wit}_y = \mathsf{NIZK.Ext}(\mathsf{t}_{\mathsf{ext},1}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_{0,1}, \mathsf{ct}_{1,1}, i_k, y), \pi_{1,k})$.
        (b) If $\mathsf{R}_{\mathcal{L}}(y, \mathsf{wit}_y) = 0$, output $\perp$.
        (c) Else, generate $\pi_1^* \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}_1, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0^*, \mathsf{ct}_1^*, (i_1 + i_2), y), \mathsf{wit}_y)$.
    8. Else, generate $\pi_1^* \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0^*, \mathsf{ct}_1^*, (i_1 + i_2), y), (\mathsf{sk}_0, \mathsf{sk}_1))$.
    9. Output $(\mathsf{ct}_0^*, \mathsf{ct}_1^*, i, \pi_0^*, \pi_1^*)$.
```

Figure 23: Circuit $\mathsf{C}_{\mathsf{Mult}}$ for the symmetric MMap

- When the element y is a non-member for the language $\mathcal{L}$, then under a binding crs_1, the "**If**" condition in step 7 of $\mathsf{C}_{\mathsf{Add}}$ is never satisfied. This again follows from the perfect soundness guarantee of the NIZK proof system. However, the condition may be satisfied during some hybrid in the proof of security, when we deliberately switch y to a member of $\mathcal{L}$.

9.5 Extraction and Zero-Testing

Extraction. We now describe the procedure for extracting a canonical string from an encoding. Suppose we have an encodings at the level i of the form:

$$(\mathsf{ct}_0, \mathsf{ct}_1, i, \pi_0, \pi_1).$$

The extraction circuit uses sk_0 and sk_1 to recover the plaintext elements $\{a_{\ell,\ell',0}, a_{\ell,\ell',1}\}_{\ell+\ell'\in[0,n]}$ underlying the FHE ciphertexts ct_0 and ct_1, and provided that these are consistent as per relation R_1 described earlier, outputs

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

Figure 24 details the operation of the extraction circuit $\mathsf{C}_{\mathsf{ext}}$. Note that it again embeds multiple secrets, including the FHE secret keys sk_0 and sk_1, as well as the extraction trapdoors $\mathsf{t}_{\mathsf{ext},0}$ and $\mathsf{t}_{\mathsf{ext},1}$ 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.

Similar to the addition, inversion and multiplication procedures described previously, the checks in steps 5 and 6 of $\mathsf{C}_{\mathsf{ext}}$ are included for technical reasons that are relevant to the proof of security. In particular, we emphasize the following:

```
$\mathsf{C}_{\mathsf{ext}}[\{\mathsf{sk}_b, \mathsf{pk}_b, \mathsf{crs}_b, \mathsf{t}_{\mathsf{ext},b}\}_{b\in\{0,1\}}, \{g_{\ell,\ell'}\}](\mathsf{ct}_0, \mathsf{ct}_1, i, \pi_0, \pi_1)$:

  1. Output $\perp$ if $\mathsf{NIZK.Verify}(\mathsf{crs}_0, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0, \mathsf{ct}_1, i, z), \pi_0) = 0$.
  2. Output $\perp$ if $\mathsf{NIZK.Verify}(\mathsf{crs}_1, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0, \mathsf{ct}_1, i, y), \pi_1) = 0$.
  3. For $b \in \{0,1\}$, recover $(\{a_{\ell,\ell',b}\}_{\ell+\ell'\in[0,n]}, i) = \mathsf{FHE.Dec}(\mathsf{sk}_b, \mathsf{ct}_b)$.
  4. Compute $g^* = \prod_{\ell+\ell'\in[0,n]} (g_{\ell,\ell'})^{a_{\ell,\ell',0}}$.
  5. If $\mathsf{R}_0((\mathsf{sk}_0, \mathsf{sk}_1), (\mathsf{ct}_0, \mathsf{ct}_1, i, \mathsf{pk}_0, \mathsf{pk}_1)) = 0$, then:
      (a) Extract $\mathsf{wit}_z = \mathsf{NIZK.Ext}(\mathsf{t}_{\mathsf{ext},0}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0, \mathsf{ct}_1, i, z), \pi_0)$.
      (b) If $\mathsf{R}_{\mathcal{L}}(z, \mathsf{wit}_z) = 0$, output $\perp$.
  6. If $\mathsf{R}_1((\mathsf{sk}_0, \mathsf{sk}_1), (\mathsf{ct}_0, \mathsf{ct}_1, i, \mathsf{pk}_0, \mathsf{pk}_1)) = 0$, then:
      (a) Extract $\mathsf{wit}_y = \mathsf{NIZK.Ext}(\mathsf{t}_{\mathsf{ext},1}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0, \mathsf{ct}_1, i, y), \pi_1)$.
      (b) If $\mathsf{R}_{\mathcal{L}}(y, \mathsf{wit}_y) = 0$, output $\perp$.
  7. Output $g^*$.
```

Figure 24: Circuit $\mathsf{C}_{\mathsf{ext}}$ for the symmetric MMap

- When the element z is a non-member for the language $\mathcal{L}$, then under a binding crs_0, the "**If**" condition in step 5 of $\mathsf{C}_{\mathsf{Inv}}$ is never satisfied. This again follows from the perfect soundness guarantee of the NIZK proof system. However, the condition may be satisfied during some hybrid in the proof of security, when some element z is a member of $\mathcal{L}$.

- When the element y is a non-member for the language $\mathcal{L}$, then under a binding crs_1, the "**If**" condition in step 6 of $\mathsf{C}_{\mathsf{Inv}}$ is never satisfied. This follows from the perfect soundness guarantee of the NIZK proof system. However, the condition may be satisfied during some hybrid in the proof of security, when the element $y \in \mathcal{X}$ is a member of $\mathcal{L}$.

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 $g \in \mathbb{G}$.

10 Proof of $(n+1)$-EDDH Hardness

In this section, we prove that solving $(n+1)$-EDDH is hard over our proposed MMap construction if solving CP-EDDH is hard over the group $\mathbb{G}$. More specifically we state and prove the following theorem:

Theorem 10.1. *The $(n+1)$-EDDH assumption holds over our proposed MMap construction provided that all of the following assumptions hold:*

- *The piO scheme is indistinguishability-secure against X-IND samplers.*
- *Membership in $\mathcal{L}$ is computationally hard to decide,*
- *Members of $\mathcal{L}$ have unique witnesses for membership.*
- *The dual-mode NIZK proof system satisfies CRS indistinguishability, perfect extractability in the binding mode, and perfect witness-indistinguishability in the hiding mode.*

Outer Hybrid	MMap Circuits	z	$\{g_{\ell,\ell'}\}$	Challenge Encodings			
				$(n+1)$-EDDH/Random	Representation	Witness for π_0	Witness for π_1
0	C_{op}	$z \notin \mathcal{L}$	Random	Random	Normal	$(\mathsf{sk}_0, \mathsf{sk}_1)$	$(\mathsf{sk}_0, \mathsf{sk}_1)$
1	$\widehat{\mathsf{C}}_{\mathsf{op}}$	$z \in \mathcal{L}$	Random	Random	Normal	wit_z	$(\mathsf{sk}_0, \mathsf{sk}_1)$
2	$\widehat{\mathsf{C}}_{\mathsf{op}}$	$z \in \mathcal{L}$	Random	Random	Partially oblique	wit_z	$(\mathsf{sk}_0, \mathsf{sk}_1)$
3	$\widehat{\mathsf{C}}_{\mathsf{op}}$	$z \in \mathcal{L}$	Random	Random	Oblique	wit_z	$(\mathsf{sk}_0, \mathsf{sk}_1)$
4	$\widehat{\mathsf{C}}_{\mathsf{op}}$	$z \in \mathcal{L}$	CP-EDDH	$(n+1)$-EDDH	Oblique	wit_z	$(\mathsf{sk}_0, \mathsf{sk}_1)$
5	$\widehat{\mathsf{C}}_{\mathsf{op}}$	$z \in \mathcal{L}$	CP-EDDH	$(n+1)$-EDDH	Partially oblique	wit_z	$(\mathsf{sk}_0, \mathsf{sk}_1)$
6	$\widehat{\mathsf{C}}_{\mathsf{op}}$	$z \in \mathcal{L}$	CP-EDDH	$(n+1)$-EDDH	Normal	wit_z	$(\mathsf{sk}_0, \mathsf{sk}_1)$
7	C_{op}	$z \notin \mathcal{L}$	CP-EDDH	$(n+1)$-EDDH	Normal	$(\mathsf{sk}_0, \mathsf{sk}_1)$	$(\mathsf{sk}_0, \mathsf{sk}_1)$
8	C_{op}	$z \notin \mathcal{L}$	Random	$(n+1)$-EDDH	Normal	$(\mathsf{sk}_0, \mathsf{sk}_1)$	$(\mathsf{sk}_0, \mathsf{sk}_1)$

Table 23: Overview of the outer hybrids in the proof of $(n+1)$-EDDH. Changes between subsequent hybrids are highlighted in red. Throughout, crs_0 and crs_1 are binding, $y \notin \mathcal{L}$, and the challenge encodings are well-formed and consistent with respect to extraction. We use the shorthands C_{op} and $\widehat{\mathsf{C}}_{\mathsf{op}}$ for the tuples $(\mathsf{C}_{\mathsf{Add}}, \mathsf{C}_{\mathsf{Inv}}, \mathsf{C}_{\mathsf{Mult}}, \mathsf{C}_{\mathsf{ext}})$ and $(\widehat{\mathsf{C}}_{\mathsf{Add}}, \widehat{\mathsf{C}}_{\mathsf{Inv}}, \widehat{\mathsf{C}}_{\mathsf{Mult}}, \widehat{\mathsf{C}}_{\mathsf{ext}})$, respectively, where the second set of circuits are described in detail subsequently.

- *The $(n+1)$-EDDH assumption holds over the group $\mathbb{G}$.*

We begin by outlining the hybrids that are used in the proof. The hybrids are classified into two broad categories - outer hybrids and inner hybrids.

10.1 Outer Hybrids

We begin by describing the outer hybrids for our proof. Outer hybrid 0 corresponds to the game where the challenger uniformly samples $\gamma, \delta \leftarrow \mathbb{Z}_q$ and provides the adversary with level-1 encodings of the form

$$[\mu]_1, [\gamma]_1, [\delta]_1,$$

where $\mu, \gamma, \delta \leftarrow \mathbb{Z}_q$ are uniformly sampled. The final outer hybrid corresponds to the game where the challenger provides the adversary with a valid $(n+1)$-EDDH instance, i.e., it uniformly samples $\mu, \gamma \leftarrow \mathbb{Z}_q$ and provides the adversary with level-1 encodings of the form

$$[\mu]_1, [\gamma]_1, [\gamma^{n+1}]_1,$$

where $\mu, \gamma \leftarrow \mathbb{Z}_q$ are uniformly sampled. Before we describe the outer hybrids, we describe a few conditions that remain invariant throughout the outer hybrids:

- The NIZK CRS strings crs_0 and crs_1 are in binding mode, as in the real MMap scheme, in all the outer hybrids.

- The element y in the public parameter is a non-member for the language $\mathcal{L}$, as in the real MMap scheme, in all the outer hybrids.
- The challenge encodings provided to the adversary are well-formed and consistent (as formalized by the relation R_1 described earlier) in all the outer hybrids. However, as we shall see later, they may be switched from the normal to oblique representation and vice-versa.

Table 23 provides an overview of the outer hybrids, which we now describe in details.

Outer Hybrid 0. In this hybrid, the MMap is set up exactly as in the real scheme described earlier. Let $(\{g_{\ell,\ell'}\})$ be the tuple of group elements hardwired into each of the MMap circuits, such that:

$$g_{\ell,\ell'} = g^{\gamma^{\ell}\cdot\delta^{\ell'}} \text{ for each } \ell,\ell' \in [0,n] \text{ such that } \ell+\ell' \leq n,$$

where $g \leftarrow \mathbb{G}$ and $\gamma,\delta \leftarrow \mathbb{Z}_q$ are uniformly sampled. Let μ be a uniformly sampled element in $\mathbb{Z}_q$. The $(n+1)$-EDDH adversary is provided with level-1 encodings of the tuple of elements:

$$(\alpha_0,\alpha_1,\alpha_2) = (\mu,\mu\cdot\gamma,\mu\cdot\delta),$$

where the encodings are generated in normal form (formalized by relation R_0 described earlier). In particular, the plaintexts underlying the FHE ciphertexts corresponding to α_0,α_1 and α_2 are of the form

$$\begin{array}{rcl} (\mu,0,0,0,\ldots,0,1) & , & (\mu,0,0,0,\ldots,0,1), \\ (\mu\cdot\gamma,0,0,0,\ldots,0,1) & , & (\mu\cdot\gamma,0,0,0,\ldots,0,1), \\ (\mu\cdot\delta,0,0,0,\ldots,0,1) & , & (\mu\cdot\delta,0,0,0,\ldots,0,1). \end{array}$$

Along with the FHE encryptions, each encoding also contains a NIZK proof π_0 for normal representation under the binding crs_0, and a NIZK proof π_1 for well-formedness and consistency with respect to extraction under the binding crs_1.

Outer Hybrid 1. In this hybrid, we make the following alterations to the manner in which the MMap is generated:

1. We switch the element z in the public parameters from a non-member to a member for $\mathcal{L}$, i.e., we now sample $z \leftarrow \mathcal{L}$, along with a (unique) membership-witness wit_z.
2. We switch the circuits for addition, inversion, multiplication and extraction as follows:

$$\begin{array}{rcl} \mathsf{C}_{\mathsf{Add}} \mapsto \widehat{\mathsf{C}}_{\mathsf{Add}} & , & \mathsf{C}_{\mathsf{Inv}} \mapsto \widehat{\mathsf{C}}_{\mathsf{Inv}}, \\ \mathsf{C}_{\mathsf{Mult}} \mapsto \widehat{\mathsf{C}}_{\mathsf{Mult}} & , & \mathsf{C}_{\mathsf{ext}} \mapsto \widehat{\mathsf{C}}_{\mathsf{ext}}, \end{array}$$

The switched circuits are described in Figures 25, 26, 27, and 28, respectively. At a high level, in each of these switched circuits, we hardwire the witness wit_z for the membership of z in $\mathcal{L}$ and avoid using the first extraction trapdoor $\mathsf{t}_{\mathsf{ext},0}$ inside the first **If** branch, which checks for normal encoding. In other words, we "allow" encodings in any level-set i' such that $i'_j = 1$ to be encoded using the oblique representation.

The following items are worth noting:

1. Any two level-sets i' and i'' such that $i'_j = i''_j = 1$ are incompatible for multiplication. This includes the adversarially chosen level set i. Hence, the multiplication circuit never evaluates $\mathsf{C}_{\mathsf{Mult,FHE}}$ to multiply two encodings that are both in the oblique representation.
2. The element y continues to be a non-member for $\mathcal{L}$ and crs_1 is still generated in the binding mode. Hence, any (adversarially) generated encoding must still satisfy consistency with respect to extraction.

Additionally, we make the following change to the manner in which the challenge encodings are generated: for each encoding, the NIZK proof π_0 under the binding crs_0 now proves that $z \in \mathcal{L}$ as opposed to proving that the encoding is in the normal representation.

```
$\widehat{\mathsf{C}}_{\mathsf{Add}}[\{\mathsf{sk}_b, \mathsf{pk}_b, \mathsf{crs}_b\}_{b\in\{0,1\}}, \mathsf{wit}_z, \mathsf{t}_{\mathsf{ext},1}, \{g_{\ell,\ell'}\}]((\mathsf{ct}_{0,1}, \mathsf{ct}_{1,1}, i, \pi_{0,1}, \pi_{1,1}), (\mathsf{ct}_{0,2}, \mathsf{ct}_{1,2}, i, \pi_{0,2}, \pi_{1,2}))$:

  1. Output $\perp$ if for any $k \in \{1,2\}$, $\mathsf{NIZK.Verify}(\mathsf{crs}_0, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_{0,k}, \mathsf{ct}_{1,k}, i, z), \pi_{0,k}) = 0$.
  2. Output $\perp$ if for any $k \in \{1,2\}$, $\mathsf{NIZK.Verify}(\mathsf{crs}_1, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_{0,k}, \mathsf{ct}_{1,k}, i, y), \pi_{1,k}) = 0$.
  3. For $b \in \{0,1\}$, set: $\mathsf{ct}_b^* = \mathsf{FHE.Eval}(\mathsf{pk}_b, \mathsf{ct}_{b,1}, \mathsf{ct}_{b,2}, \mathsf{C}_{\mathsf{Add},\mathrm{FHE}})$.
  4. For $b \in \{0,1\}$ and $k \in \{1,2\}$, recover $(\{a_{\ell,\ell',b,k}\}_{\ell+\ell'\in[0,n]}, i) = \mathsf{FHE.Dec}(\mathsf{sk}_b, \mathsf{ct}_{b,k})$.
  5. If for some $k \in \{1,2\}$, $\mathsf{R}_0((\mathsf{sk}_0, \mathsf{sk}_1), (\mathsf{ct}_{0,k}, \mathsf{ct}_{1,k}, i, \mathsf{pk}_0, \mathsf{pk}_1)) = 0$, then:
      (a) // Omitted, depends on $\mathsf{t}_{\mathsf{ext},0}$.
      (b) Generate $\pi_0^* \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}_0, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0^*, \mathsf{ct}_1^*, i, z), \mathsf{wit}_z)$.
  6. Else, generate $\pi_0^* \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0^*, \mathsf{ct}_1^*, i, z), (\mathsf{sk}_0, \mathsf{sk}_1))$.
  7. If for some $k \in \{1,2\}$, we have $\mathsf{R}_1((\mathsf{sk}_0, \mathsf{sk}_1), (\mathsf{ct}_{0,k}, \mathsf{ct}_{1,k}, i, \mathsf{pk}_0, \mathsf{pk}_1)) = 0$, then:
      (a) Extract $\mathsf{wit}_y = \mathsf{NIZK.Ext}(\mathsf{t}_{\mathsf{ext},1}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_{0,1}, \mathsf{ct}_{1,1}, i, y), \pi_{1,1})$.
      (b) If $\mathsf{R}_{\mathcal{L}}(y, \mathsf{wit}_y) = 0$, output $\perp$.
      (c) Else, generate $\pi_1^* \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}_1, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0^*, \mathsf{ct}_1^*, i, y), \mathsf{wit}_y)$.
  8. Else, generate $\pi_1^* \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0^*, \mathsf{ct}_1^*, i, y), (\mathsf{sk}_0, \mathsf{sk}_1))$.
  9. Output $(\mathsf{ct}_0^*, \mathsf{ct}_1^*, i, \pi_0^*, \pi_1^*)$.
```

Figure 25: Circuit $\widehat{\mathsf{C}}_{\mathsf{Add}}$ for the symmetric MMap

Outer Hybrid 2. This hybrid is identical to the outer hybrid 1, except that the challenge $(n+1)$-EDDH encodings are no longer in normal form. In particular, the first FHE ciphertext in each encoding now encrypts an oblique form of the underlying plaintext element. The representations of the plaintexts corresponding to α_0,α_1 and α_2 are of the form

$$\begin{array}{ccc} (\mu, 0, 0, 0, \ldots, 0, 1) & , & (\mu, 0, 0, 0, \ldots, 0, 1), \\ (0, \mu, 0, 0, \ldots, 0, 1) & , & (\mu \cdot \gamma, 0, 0, 0, \ldots, 0, 1), \\ (0, 0, \mu, 0, \ldots, 0, 1) & , & (\mu \cdot \delta, 0, 0, 0, \ldots, 0, 1). \end{array}$$

where the non-zero entries in the second plaintext correspond to the indices $(\ell, \ell') = (0,0), (1,0), (0,1)$, respectively. We represent the encodings as a vector with these as the first three indices for ease of representation. As in outer hybrid 1, each encoding also contains a NIZK proof π_0 for $z \in \mathcal{L}$ under the binding crs_0, and a NIZK proof π_1 for well-formedness and consistency with respect to extraction under the binding crs_1.

Outer Hybrid 3. This hybrid is identical to the outer hybrid 2, except that the challenge $(n+1)$-EDDH encodings are now entirely in oblique form. In particular, the representations of the plaintexts corresponding to α_0,α_1 and α_2 are of the form

$$\begin{array}{ccc} (\mu, 0, 0, 0, \ldots, 0, 1) & , & (\mu, 0, 0, 0, \ldots, 0, 1), \\ (0, \mu, 0, 0, \ldots, 0, 1) & , & (0, \mu, 0, 0, \ldots, 0, 1), \\ (0, 0, \mu, 0, \ldots, 0, 1) & , & (0, 0, \mu, 0, \ldots, 0, 1). \end{array}$$

where the non-zero entries in both plaintexts correspond to the indices $(\ell, \ell') = (0,0), (1,0), (0,1)$, respectively. Each encoding continues to have a NIZK proof π_0 for $z \in \mathcal{L}$ under the binding crs_0, and a NIZK proof π_1 for well-formedness and consistency with respect to extraction under the binding crs_1.

```
$\widehat{\mathsf{C}}_{\mathsf{Inv}}[\{\mathsf{sk}_b, \mathsf{pk}_b, \mathsf{crs}_b\}_{b\in\{0,1\}}, \mathsf{wit}_z, \mathsf{t}_{\mathsf{ext},1}, \{g_{\ell,\ell'}\}](\mathsf{ct}_0, \mathsf{ct}_1, i, \pi_0, \pi_1)$:

1. Output $\perp$ if $\mathsf{NIZK.Verify}(\mathsf{crs}_0, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0, \mathsf{ct}_1, i, z), \pi_0) = 0$.

2. Output $\perp$ if $\mathsf{NIZK.Verify}(\mathsf{crs}_1, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0, \mathsf{ct}_1, i, y), \pi_1) = 0$.

3. For $b \in \{0,1\}$, set: $\mathsf{ct}_b^* = \mathsf{FHE.Eval}(\mathsf{pk}_b, \mathsf{ct}_b, \mathsf{C}_{\mathsf{Add,FHE}})$.

4. For $b \in \{0,1\}$, recover $(\{a_{\ell,\ell',b}\}_{\ell+\ell'\in[0,n]}, i) = \mathsf{FHE.Dec}(\mathsf{sk}_b, \mathsf{ct}_b)$.

5. If $\mathsf{R}_0((\mathsf{sk}_0, \mathsf{sk}_1), (\mathsf{ct}_0, \mathsf{ct}_1, i, \mathsf{pk}_0, \mathsf{pk}_1)) = 0$, then:

    (a) // Omitted, depends on $\mathsf{t}_{\mathsf{ext},0}$.

    (b) Generate $\pi_0^* \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}_0, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0^*, \mathsf{ct}_1^*, i, z), \mathsf{wit}_z)$.

6. Else, generate $\pi_0^* \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0^*, \mathsf{ct}_1^*, i, z), (\mathsf{sk}_0, \mathsf{sk}_1))$.

7. If $\mathsf{R}_1((\mathsf{sk}_0, \mathsf{sk}_1), (\mathsf{ct}_0, \mathsf{ct}_1, i, \mathsf{pk}_0, \mathsf{pk}_1)) = 0$, then:

    (a) Extract $\mathsf{wit}_y = \mathsf{NIZK.Ext}(\mathsf{t}_{\mathsf{ext},1}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0, \mathsf{ct}_1, i, y), \pi_1)$.

    (b) If $\mathsf{R}_{\mathcal{L}}(y, \mathsf{wit}_y) = 0$, output $\perp$.

    (c) Else, generate $\pi_1^* \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}_1, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0^*, \mathsf{ct}_1^*, i, y), \mathsf{wit}_y)$.

8. Else, generate $\pi_1^* \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0^*, \mathsf{ct}_1^*, i, y), (\mathsf{sk}_0, \mathsf{sk}_1))$.

9. Output $(\mathsf{ct}_0^*, \mathsf{ct}_1^*, i, \pi_0^*, \pi_1^*)$.
```

Figure 26: Circuit $\widehat{\mathsf{C}}_{\mathsf{Inv}}$ for the symmetric MMap

Outer Hybrid 4. This hybrid is identical to outer hybrid 3 except that we switch the tuple of group elements $(\{g_{\ell,\ell'}\})$ hardwired into the modified MMap circuits $\widehat{\mathsf{C}}_{\mathsf{Add}}$, $\widehat{\mathsf{C}}_{\mathsf{Inv}}$, $\widehat{\mathsf{C}}_{\mathsf{Mult}}$ and $\widehat{\mathsf{C}}_{\mathsf{ext}}$ to a valid CP-EDDH tuple. More formally, we hardwire a tuple of group elements $(\{g_{\ell,\ell'}\})$ into each of the modified MMap circuits, such that:

$$g_{\ell,\ell'} = g^{\gamma^{\ell+(n+1)\cdot\ell'}} \text{ for each } \ell, \ell' \in [0,n] \text{ such that } \ell + \ell' \leq n,$$

where $g \leftarrow \mathbb{G}$ and $\gamma \leftarrow \mathbb{Z}_q$ are uniformly sampled.

Outer Hybrid 5. This hybrid is identical to the outer hybrid 4, except that the challenge $(n+1)$-EDDH encodings are now switched back to a partially oblique form. In particular, the representations of the plaintexts corresponding to α_0,α_1 and α_2 are of the form

$$\begin{array}{ccc} (\mu, 0, 0, 0, \ldots, 0, 1) & , & (\mu, 0, 0, 0, \ldots, 0, 1), \\ (0, \mu, 0, 0, \ldots, 0, 1) & , & (\mu \cdot \gamma, 0, 0, 0, \ldots, 0, 1), \\ (0, 0, \mu, 0, \ldots, 0, 1) & , & (\mu \cdot \gamma^{n+1}, 0, 0, 0, \ldots, 0, 1). \end{array}$$

where the non-zero entries in both plaintexts correspond to the indices $(\ell, \ell') = (0,0), (1,0), (0,1)$, respectively. Each encoding continues to have a NIZK proof π_0 for $z \in \mathcal{L}$ under the binding crs_0, and a NIZK proof π_1 for well-formedness and consistency with respect to extraction under the binding crs_1.

Outer Hybrid 6. This hybrid is identical to the outer hybrid 5, except that the challenge $(n+1)$-EDDH encodings are now switched back entirely to the normal form. In particular, the representations of the plaintexts corresponding

```
$\widehat{\mathsf{C}}_{\mathsf{Mult}}[\{\mathsf{sk}_b, \mathsf{pk}_b, \mathsf{crs}_b\}_{b\in\{0,1\}}, \mathsf{wit}_z, \mathsf{t}_{\mathsf{ext},1}, \{g_{\ell,\ell'}\}]((\mathsf{ct}_{0,1}, \mathsf{ct}_{1,1}, i_1, \pi_{0,1}, \pi_{1,1}), (\mathsf{ct}_{0,2}, \mathsf{ct}_{1,2}, i_2, \pi_{0,2}, \pi_{1,2}))$:

1. Output $\perp$ if for any $k \in \{1,2\}$, $\mathsf{NIZK.Verify}(\mathsf{crs}_0, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_{0,k}, \mathsf{ct}_{1,k}, i_k, z), \pi_{0,k}) = 0$.
2. Output $\perp$ if for any $k \in \{1,2\}$, $\mathsf{NIZK.Verify}(\mathsf{crs}_1, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_{0,k}, \mathsf{ct}_{1,k}, i_k, y), \pi_{1,k}) = 0$.
3. For $b \in \{0,1\}$, set: $\mathsf{ct}_b^* = \mathsf{FHE.Eval}(\mathsf{pk}_b, \mathsf{ct}_{b,1}, \mathsf{ct}_{b,2}, \mathsf{C}_{\mathsf{Mult,FHE}})$.
4. For $b \in \{0,1\}$ and $k \in \{1,2\}$, recover $(\{a_{\ell,\ell',b,k}\}_{\ell+\ell'\in[0,n]}, i_k) = \mathsf{FHE.Dec}(\mathsf{sk}_b, \mathsf{ct}_{b,k})$.
5. If for some $k \in \{1,2\}$, $\mathsf{R}_0((\mathsf{sk}_0, \mathsf{sk}_1), (\mathsf{ct}_{0,k}, \mathsf{ct}_{1,k}, i_k, \mathsf{pk}_0, \mathsf{pk}_1)) = 0$, then:
    (a) // Omitted, depends on $\mathsf{t}_{\mathsf{ext},0}$.
    (b) Generate $\pi_0^* \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}_0, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0^*, \mathsf{ct}_1^*, (i_1 + i_2), z), \mathsf{wit}_z)$.
6. Else, generate $\pi_0^* \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0^*, \mathsf{ct}_1^*, (i_1 + i_2), z), (\mathsf{sk}_0, \mathsf{sk}_1))$.
7. If for some $k \in \{1,2\}$, we have $\mathsf{R}_1((\mathsf{sk}_0, \mathsf{sk}_1), (\mathsf{ct}_{0,k}, \mathsf{ct}_{1,k}, i_k, \mathsf{pk}_0, \mathsf{pk}_1)) = 0$, then:
    (a) Extract $\mathsf{wit}_y = \mathsf{NIZK.Ext}(\mathsf{t}_{\mathsf{ext},1}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_{0,1}, \mathsf{ct}_{1,1}, i_k, y), \pi_{1,k})$.
    (b) If $\mathsf{R}_{\mathcal{L}}(y, \mathsf{wit}_y) = 0$, output $\perp$.
    (c) Else, generate $\pi_1^* \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}_1, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0^*, \mathsf{ct}_1^*, (i_1 + i_2), y), \mathsf{wit}_y)$.
8. Else, generate $\pi_1^* \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0^*, \mathsf{ct}_1^*, (i_1 + i_2), y), (\mathsf{sk}_0, \mathsf{sk}_1))$.
9. Output $(\mathsf{ct}_0^*, \mathsf{ct}_1^*, i, \pi_0^*, \pi_1^*)$.
```

Figure 27: Circuit $\widehat{\mathsf{C}}_{\mathsf{Mult}}$ for the symmetric MMap

```
$\widehat{\mathsf{C}}_{\mathsf{ext}}[\{\mathsf{sk}_b, \mathsf{pk}_b, \mathsf{crs}_b\}_{b\in\{0,1\}}, \mathsf{wit}_z, \mathsf{t}_{\mathsf{ext},1}, \{g_{\ell,\ell'}\}](\mathsf{ct}_0, \mathsf{ct}_1, i, \pi_0, \pi_1)$:

1. Output $\perp$ if $\mathsf{NIZK.Verify}(\mathsf{crs}_0, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0, \mathsf{ct}_1, i, z), \pi_0) = 0$.
2. Output $\perp$ if $\mathsf{NIZK.Verify}(\mathsf{crs}_1, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0, \mathsf{ct}_1, i, y), \pi_1) = 0$.
3. For $b \in \{0,1\}$, recover $(\{a_{\ell,\ell',b}\}_{\ell+\ell'\in[0,n]}, i) = \mathsf{FHE.Dec}(\mathsf{sk}_b, \mathsf{ct}_b)$.
4. Compute $g^* = \prod_{\ell+\ell'\in[0,n]} (g_{\ell,\ell'})^{a_{\ell,\ell',0}}$.
5. // Omitted, depends on $\mathsf{t}_{\mathsf{ext},0}$.
6. If $\mathsf{R}_1((\mathsf{sk}_0, \mathsf{sk}_1), (\mathsf{ct}_0, \mathsf{ct}_1, i, \mathsf{pk}_0, \mathsf{pk}_1)) = 0$, then:
    (a) Extract $\mathsf{wit}_y = \mathsf{NIZK.Ext}(\mathsf{t}_{\mathsf{ext},1}, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_0, \mathsf{ct}_1, i, y), \pi_1)$.
    (b) If $\mathsf{R}_{\mathcal{L}}(y, \mathsf{wit}_y) = 0$, output $\perp$.
7. Output $g^*$.
```

Figure 28: Circuit $\widehat{\mathsf{C}}_{\mathsf{ext}}$ for the symmetric MMap

to α_0,α_1 and α_2 are of the form

$$\begin{aligned}(\mu, 0, 0, 0, \ldots, 0, 1) \quad &, \quad (\mu, 0, 0, 0, \ldots, 0, 1),\\ (\mu \cdot \gamma, 0, 0, 0, \ldots, 0, 1) \quad &, \quad (\mu \cdot \gamma, 0, 0, 0, \ldots, 0, 1),\\ (\mu \cdot \gamma^{n+1}, 0, 0, 0, \ldots, 0, 1) \quad &, \quad (\mu \cdot \gamma^{n+1}, 0, 0, 0, \ldots, 0, 1).\end{aligned}$$

where the non-zero entries in both plaintexts correspond to the indices $(\ell, \ell') = (0,0), (1,0), (0,1)$, respectively. Each encoding continues to have a NIZK proof π_0 for $z \in \mathcal{L}$ under the binding crs_0, and a NIZK proof π_1 for well-formedness and consistency with respect to extraction under the binding crs_1.

Outer Hybrid 7. This hybrid is identical to outer hybrid 5, except that we make the following alterations to the manner in which the MMap is set up:

1. We switch back the element z in the public parameters from a member to a non-member for $\mathcal{L}$, i.e., we now sample $z \leftarrow \mathcal{X} \setminus \mathcal{L}$. Note that this is exactly as in the real MMap scheme.

2. We switch back the circuits for addition, inversion, multiplication and extraction as follows:

$$\begin{array}{ccc} \widehat{\mathsf{C}}_{\mathsf{Add}} \mapsto \mathsf{C}_{\mathsf{Add}} & , & \widehat{\mathsf{C}}_{\mathsf{Inv}} \mapsto \mathsf{C}_{\mathsf{Inv}}, \\ \widehat{\mathsf{C}}_{\mathsf{Mult}} \mapsto \mathsf{C}_{\mathsf{Mult}} & , & \widehat{\mathsf{C}}_{\mathsf{ext}} \mapsto \mathsf{C}_{\mathsf{ext}}, \end{array}$$

   In particular, the circuits are now exactly as in the real MMap scheme, with the exception that the tuple $(\{g_{\ell,\ell'}\})$ hardwired inside these circuits continues to be a CP-EDDH tuple.

Additionally, we make the following change to the manner in which the challenge encodings are generated: for each encoding, the NIZK proof π_0 under the binding crs_0 now proves that the encoding is in the normal representation using the witness $(\mathsf{sk}_0, \mathsf{sk}_1)$, as opposed to proving that $z \in \mathcal{L}$.

Outer Hybrid 8. This hybrid is identical to outer hybrid 7 except that we switch the tuple of group elements $(\{g_{\ell,\ell'}\})$ hardwired into the MMap circuits $\mathsf{C}_{\mathsf{Add}}$, $\mathsf{C}_{\mathsf{Inv}}$, $\mathsf{C}_{\mathsf{Mult}}$ and $\mathsf{C}_{\mathsf{ext}}$ from a uniform CP-EDDH tuple back to a tuple as in the real scheme. More formally, we hardwire a tuple of group elements $(\{g_{\ell,\ell'}\})$ into each of the modified MMap circuits, such that:

$$g_{\ell,\ell'} = g^{\gamma^{\ell} \cdot \delta^{\ell'}} \text{ for each } \ell, \ell' \in [0,n] \text{ such that } \ell + \ell' \leq n,$$

where $g \leftarrow \mathbb{G}$ and $\gamma, \delta \leftarrow \mathbb{Z}_q$ are uniformly sampled.

10.2 Indistinguishability of Outer Hybrids

In this section, we argue that the outer hybrids are computationally indistinguishable from each other. Each argument in turn involves a sequence of inner hybrids, as described below.

Outer Hybrids 0 and 1. We first argue that outer hybrids 0 and 1 are computationally indistinguishable from each other. In particular, we state and prove the following lemma:

Lemma 10.2. *The outer hybrids 0 and 1 are computationally indistinguishable provided that all of the following assumptions hold:*

- *The piO scheme is indistinguishability-secure against X-IND samplers.*
- *Membership in $\mathcal{L}$ is computationally hard to decide,*
- *Members of $\mathcal{L}$ have unique witnesses for membership.*
- *The dual-mode NIZK proof system satisfies CRS indistinguishability, perfect extractability in the binding mode, and perfect witness-indistinguishability in the hiding mode.*

Proof. The proof of this lemma uses a sequence of inner hybrids that are technically very similar to those used in the proof of Lemma 6.2. Hence, we do not detail them.

Outer Hybrids 1 and 2. We now argue that outer hybrids 1 and 2 are computationally indistinguishable from each other. In particular, we state and prove the following lemma:

Lemma 10.3. *The outer hybrids 1 and 2 are computationally indistinguishable provided that all of the following assumptions hold:*

- *The piO scheme is indistinguishability-secure against X-IND samplers.*
- *Membership in $\mathcal{L}$ is computationally hard to decide,*
- *Members of $\mathcal{L}$ have unique witnesses for membership.*
- *The dual-mode NIZK proof system satisfies CRS indistinguishability, perfect extractability in the binding mode, and perfect witness-indistinguishability in the hiding mode.*

Proof. The proof of this lemma uses a sequence of inner hybrids that are technically very similar to those used in the proof of Lemma 6.3. Hence, we do not detail them.

Outer Hybrids 2 and 3. We now argue that outer hybrids 2 and 3 are computationally indistinguishable from each other. In particular, we state and prove the following lemma:

Lemma 10.4. *The outer hybrids 2 and 3 are computationally indistinguishable provided that all of the following assumptions hold:*

- *The piO scheme is indistinguishability-secure against X-IND samplers.*
- *Membership in $\mathcal{L}$ is computationally hard to decide,*
- *Members of $\mathcal{L}$ have unique witnesses for membership.*
- *The dual-mode NIZK proof system satisfies CRS indistinguishability, perfect extractability in the binding mode, and perfect witness-indistinguishability in the hiding mode.*

Proof. The proof of this lemma uses a sequence of inner hybrids that are technically very similar to those used in the proof of Lemma 6.6. Hence, we do not detail them.

Outer Hybrids 3 and 4. We state and prove the following lemma:

Lemma 10.5. *The outer hybrids 3 and 4 are computationally indistinguishable provided that the CP-EDDH assumption holds over the group $\mathbb{G}$.*

Proof. Suppose that there exists a PPT adversary $\mathcal{A}$ that can distinguish between the outer hybrids 3 and 4 with non-negligible probability. We construct a PPT algorithm $\mathcal{B}$ that can break the CP-EDDH assumption over the group $\mathbb{G}$ with non-negligible probability. As part of its input CP-EDDH challenge, $\mathcal{B}$ receives a tuple of group elements of the form $\{g_{\ell,\ell'}\}$, and sets up the public parameters for the MMap as follows:

1. $\mathcal{B}$ samples two key-pairs for the FHE scheme as:

$$(\mathsf{pk}_0, \mathsf{sk}_0), (\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathsf{FHE.Gen}(1^\lambda),$$

   and a pair of binding NIZK CRS strings (along with the corresponding extraction trapdoors) as:

$$(\mathsf{crs}_0, \mathsf{t}_{\mathsf{ext},0}), (\mathsf{crs}_1, \mathsf{t}_{\mathsf{ext},1}) \leftarrow \mathsf{NIZK.Setup}(1^\lambda, \mathsf{binding}).$$

2. Next $\mathcal{B}$ uniformly samples
$$y \leftarrow \mathcal{X} \setminus \mathcal{L} \quad , \quad z \leftarrow \mathcal{L},$$
where z has unique membership-witness wit_z.

3. Finally, $\mathcal{B}$ sets up the MMap circuits $\widehat{\mathsf{C}}_{\mathsf{Add}}$, $\widehat{\mathsf{C}}_{\mathsf{Inv}}$, $\widehat{\mathsf{C}}_{\mathsf{Mult}}$ and $\widehat{\mathsf{C}}_{\mathsf{ext}}$ exactly as described in Figures 25, 26, 27 and 28, respectively, except for the fact that it hardwires its input tuple $\{g_{\ell,\ell'}\}$ into each of these circuits. It then computes and outputs four probabilistically indistinguishability-obfuscated circuits of the form
$$\bar{\mathsf{C}}_{\mathsf{Add}} = \mathsf{piO.Obf}(\widehat{\mathsf{C}}_{\mathsf{Add}}) \quad , \quad \bar{\mathsf{C}}_{\mathsf{Inv}} = \mathsf{piO.Obf}(\widehat{\mathsf{C}}_{\mathsf{Inv}}),$$
$$\bar{\mathsf{C}}_{\mathsf{Mult}} = \mathsf{piO.Obf}(\widehat{\mathsf{C}}_{\mathsf{Mult}}) \quad , \quad \bar{\mathsf{C}}_{\mathsf{ext}} = \mathsf{piO.Obf}(\widehat{\mathsf{C}}_{\mathsf{ext}}).$$

Next, $\mathcal{B}$ sets up the challenge encodings in the oblique representation as follows:

1. $\mathcal{B}$ samples $\mu \leftarrow \mathbb{Z}_q$ and generates the following FHE ciphertexts for each $b \in \{0, 1\}$:
$$\mathsf{ct}_{b,0} = \mathsf{FHE.Enc}(\mathsf{pk}_b, (\mu, 0, 0, 0 \ldots, 0, 1)),$$
$$\mathsf{ct}_{b,1} = \mathsf{FHE.Enc}(\mathsf{pk}_b, (0, \mu, 0, 0 \ldots, 0, 1),$$
$$\mathsf{ct}_{b,2} = \mathsf{FHE.Enc}(\mathsf{pk}_b, (0, 0, \mu, 0, \ldots, 0, 1).$$

2. $\mathcal{B}$ generates the following proofs for each $\ell \in \{0, 1, 2\}$:
$$\begin{aligned}\pi_{0,\ell} &= \mathsf{NIZK.Prove}(\mathsf{crs}_0, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_{0,\ell}, \mathsf{ct}_{1,\ell}, 1, \mathsf{wit}_z),\\ \pi_{1,\ell} &= \mathsf{NIZK.Prove}(\mathsf{crs}_1, (\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ct}_{0,\ell}, \mathsf{ct}_{1,\ell}, 1, y), (\mathsf{sk}_0, \mathsf{sk}_1)).\end{aligned}$$

3. Finally, for each $\ell \in \{0, 1, 2\}$, $\mathcal{B}$ generates the encodings for α_ℓ as:
$$[\alpha_\ell] = \Big(\mathsf{ct}_{0,\ell}, \mathsf{ct}_{1,\ell}, 1, \pi_{0,\ell}, \pi_{1,\ell}\Big).$$

$\mathcal{B}$ then provides $\mathcal{A}$ with the MMap public parameters and the challenge encodings. Eventually, $\mathcal{A}$ outputs a bit $b^\star$. $\mathcal{B}$ outputs the same bit $b^\star$. Now, observe the following:

- Suppose that $\mathcal{B}$ receives as input a tuple of uniformly random group elements $\{g_{\ell,\ell'}\}$ such that
$$g_{\ell,\ell'} = g^{\gamma^{\ell} \cdot \delta^{\ell'}} \text{ for each } \ell, \ell' \in [0, n] \text{ such that } \ell + \ell' \leq n,$$
where $g \leftarrow \mathbb{G}$ and $\gamma, \delta \leftarrow \mathbb{Z}_q$ are uniformly sampled. In this case, the view of $\mathcal{A}$ is exactly as in outer hybrid 3.

- On the other hand, suppose that $\mathcal{B}$ receives as input a CP-EDDH tuple of the form $\{g_{\ell,\ell'}\}$ such that
$$g_{\ell,\ell'} = g^{\gamma^{\ell+(n+1)\cdot\ell'}} \text{ for each } \ell, \ell' \in [0, n] \text{ such that } \ell + \ell' \leq n,$$
where $g \leftarrow \mathbb{G}$ and $\gamma \leftarrow \mathbb{Z}_q$ are uniformly sampled. In this case, the view of $\mathcal{A}$ is exactly as in outer hybrid 4. □

Hence the advantage of $\mathcal{B}$ in breaking CP-EDDH over the group $\mathbb{G}$ is the same as the advantage of $\mathcal{A}$ in distinguishing the outer hybrids 3 and 4. This concludes the proof of Lemma 10.5.

Outer Hybrids 4 and 5. We state and prove the following lemma:

Lemma 10.6. *The outer hybrids 2 and 3 are computationally indistinguishable provided that all of the following assumptions hold:*

- *The piO scheme is indistinguishability-secure against X-IND samplers.*
- *Membership in $\mathcal{L}$ is computationally hard to decide,*
- *Members of $\mathcal{L}$ have unique witnesses for membership.*
- *The dual-mode NIZK proof system satisfies CRS indistinguishability, perfect extractability in the binding mode, and perfect witness-indistinguishability in the hiding mode.*

Proof. The proof of this lemma uses a sequence of inner hybrids that are technically very similar to those used in the proof of Lemma 6.8. Hence, we do not detail them.

Outer Hybrids 5 and 6. We state the following lemma:

Lemma 10.7. *The outer hybrids 5 and 6 are computationally indistinguishable provided that all of the following assumptions hold:*

- *The piO scheme is indistinguishability-secure against X-IND samplers.*
- *Membership in $\mathcal{L}$ is computationally hard to decide,*
- *Members of $\mathcal{L}$ have unique witnesses for membership.*
- *The dual-mode NIZK proof system satisfies CRS indistinguishability, perfect extractability in the binding mode, and perfect witness-indistinguishability in the hiding mode.*

Proof. The proof of this lemma uses a sequence of inner hybrids that are technically very similar to those used in the proof of Lemma 6.9. Hence, we do not detail them.

Outer Hybrids 6 and 7. We state the following lemma:

Lemma 10.8. *The outer hybrids 6 and 7 are computationally indistinguishable provided that all of the following assumptions hold:*

- *The piO scheme is indistinguishability-secure against X-IND samplers.*
- *Membership in $\mathcal{L}$ is computationally hard to decide,*
- *Members of $\mathcal{L}$ have unique witnesses for membership.*
- *The dual-mode NIZK proof system satisfies CRS indistinguishability, perfect extractability in the binding mode, and perfect witness-indistinguishability in the hiding mode*

Proof. The proof of this lemma uses a sequence of inner hybrids that are technically very similar to those used in the proof of Lemma 6.10. Hence, we do not detail them.

Outer Hybrids 7 and 8. We state and prove the following lemma:

Lemma 10.9. *The outer hybrids 7 and 8 are computationally indistinguishable provided that all of the following assumptions hold:*

- *The piO scheme is indistinguishability-secure against X-IND samplers.*
- *The dual-mode NIZK proof system satisfies perfect soundness in the binding mode.*

Proof. In this hybrid, we switch the tuple of group elements $(\{g_{\ell,\ell'}\})$ hardwired into the MMap circuits $\mathsf{C}_{\mathsf{Add}}$, $\mathsf{C}_{\mathsf{Inv}}$, $\mathsf{C}_{\mathsf{Mult}}$ and $\mathsf{C}_{\mathsf{ext}}$ from a uniform CP-EDDH tuple to a uniformly random tuple, albeit with the same base element $g_{0,0}$. We argue below that this switch does not alter the output distribution of these circuits from outer hybrid 7 to outer hybrid 8. Once this is established, the indistinguishability argument follows immediately under the assumption that the piO scheme is indistinguishability-secure against X-IND samplers.

We first focus on the MMap circuits $\mathsf{C}_{\mathsf{Add}}$, $\mathsf{C}_{\mathsf{Inv}}$ and $\mathsf{C}_{\mathsf{Mult}}$. Observe that switching $(\{g_{\ell,\ell'}\})$ from a uniform CP-EDDH tuple to a tuple distributed as in the real scheme only (potentially)affects the outcome of the "**If**" condition in step 7 of each of these circuits. Note however that in both outer hybrids 7 and 8, crs_0 is in binding mode and the element z in the public parameter is a non-member for $\mathcal{L}$. Hence, it follows from the perfect soundness of the dual-mode NIZK proof system that any input encoding(s) for which these circuits do not output $\perp$ and terminate before step 7 must be in normal representation *and* well-formed *and* consistent. This in turn guarantees that the outcome of the "**If**" condition must be "true". Hence, the output distributions of the MMap circuits $\mathsf{C}_{\mathsf{Add}}$, $\mathsf{C}_{\mathsf{Inv}}$ and $\mathsf{C}_{\mathsf{Mult}}$ in outer hybrids 7 and 8 are identical.

Finally, we focus on the MMap extraction circuit $\mathsf{C}_{\mathsf{ext}}$. Observe that switching $(\{g_{\ell,\ell'}\})$ from a uniform CP-EDDH tuple to a tuple distributed as in the real scheme potentially affects the output of the extraction circuit. However, note yet again that in both outer hybrids 7 and 8, crs_0 is in binding mode and the element z in the public parameter is a non-member for $\mathcal{L}$. Hence, it follows from the perfect soundness of the dual-mode NIZK proof system that any input encoding(s) for which the circuit does not output $\perp$ and terminate before the extraction step is executed must be in normal representation *and* well-formed *and* consistent. Observe also that the base element g_0 in the tuple of group elements remains unaltered across outer hybrids 7 and 8. It follows immediately that the outcome of extraction on a given encoding in normal representation in both hybrids is identical. In other words, the output distributions of $\mathsf{C}_{\mathsf{ext}}$ in outer hybrids 7 and 8 are identical. This concludes the proof of Lemma 10.9, and hence the proof of Theorem 10.1. □

Acknowledgements

We thank Dennis Hofheinz and Kenneth G. Paterson for many useful discussions and inputs. We thank the anonymous reviewers of TCC 2020 for pointing out a technical problem in an earlier version of the paper. The current version fixes this problem.

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.

[ABBC10] Tolga Acar, Mira Belenkiy, Mihir Bellare, and David Cash. Cryptographic agility and its relation to circular encryption. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 403–422. Springer, Heidelberg, May / June 2010.

[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.

[AH18] Thomas Agrikola and Dennis Hofheinz. Interactively secure groups from obfuscation. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018, Part II*, volume 10770 of *LNCS*, pages 341–370. Springer, Heidelberg, March 2018.

[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.

[Ale03] Michael Alekhnovich. More on average case vs approximation complexity. In *44th FOCS*, pages 298–307. IEEE Computer Society Press, October 2003.

[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.

[BB04] Dan Boneh and Xavier Boyen. Efficient selective-ID secure identity based encryption without random oracles. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 223–238. Springer, Heidelberg, May 2004.

[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.

[BDGM20a] 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.

[BDGM20b] Zvika Brakerski, Nico Döttling, Sanjam Garg, and Giulio Malavolta. Factoring and pairings are not necessary for io: Circular-secure lwe suffices. Cryptology ePrint Archive, Report 2020/1024, 2020. `https://eprint.iacr.org/2020/1024`.

[BGdMM05] Lucas Ballard, Matthew Green, Breno de Medeiros, and Fabian Monrose. Correlation-resistant storage via keyword-searchable encryption. Cryptology ePrint Archive, Report 2005/417, 2005. `https://eprint.iacr.org/2005/417`.

[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`.

[BMZ19] James Bartusek, Fermi Ma, and Mark Zhandry. The distinction between fixed and random generators in group-based assumptions. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part II*, volume 11693 of *LNCS*, pages 801–830. Springer, Heidelberg, August 2019.

[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.

[BS03] Dan Boneh and Alice Silverberg. Applications of multilinear forms to cryptography. *Contemporary Mathematics*, 324(1):71–90, 2003.

[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.

[CLL$^+$13] Jie Chen, Hoon Wei Lim, San Ling, Huaxiong Wang, and Hoeteck Wee. Shorter IBE and signatures via asymmetric pairings. In Michel Abdalla and Tanja Lange, editors, *PAIRING 2012*, volume 7708 of *LNCS*, pages 122–140. Springer, Heidelberg, May 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.

[DJ01] Ivan Damgård and Mats Jurik. A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In Kwangjo Kim, editor, *PKC 2001*, volume 1992 of *LNCS*, pages 119–136. Springer, Heidelberg, February 2001.

[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`.

[GP20] Romain Gay and Rafael Pass. Indistinguishability obfuscation from circular security. Cryptology ePrint Archive, Report 2020/1010, 2020. `https://eprint.iacr.org/2020/1010`.

[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.

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 75–92. Springer, Heidelberg, August 2013.

[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.

[JLS20] Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from well-founded assumptions. Cryptology ePrint Archive, Report 2020/1003, 2020. `https://eprint.iacr.org/2020/1003`.

[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.

[Lin17] Huijia Lin. Indistinguishability obfuscation from SXDH on 5-linear maps and locality-5 PRGs. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 599–629. Springer, Heidelberg, August 2017.

[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`.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th ACM STOC*, pages 84–93. ACM Press, May 2005.

[Rot13] Ron Rothblum. On the circular security of bit-encryption. In Amit Sahai, editor, *TCC 2013*, volume 7785 of *LNCS*, pages 579–598. Springer, Heidelberg, March 2013.

[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.