Ring Key-Homomorphic Weak PRFs and Applications

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

Abstract

A *weak* pseudorandom function $F : \mathcal{K} \times \mathcal{X} \rightarrow \mathcal{Y}$ is said to be *ring* key-homomorphic if, given $F(k_1, x)$ and $F(k_2, x)$, there are efficient algorithms to compute $F(k_1 \oplus k_2, x)$ and $F(k_1 \otimes k_2, x)$ where $\oplus$ and $\otimes$ are the addition and multiplication operations in the ring $\mathcal{K}$, respectively. In this work, we initiate the study of ring key-homomorphic weak PRFs (RKHwPRFs). In particular, we show that the following primitives can be constructed from *any* RKHwPRF:

- Multiparty non-interactive key exchange (NIKE) for an arbitrary number of parties.
- Indistinguishability obfuscation for all circuits in $\mathcal{NC}^1$.

Our proofs are in the standard model, and the proof for our iO scheme is program-independent. Our iO scheme can also be bootstrapped to all polynomial-size circuits using standard techniques. We also consider restricted versions of RKHwPRFs that are structurally weaker than a classic RKHwPRF but suffice for all our constructions. We show how to instantiate these restricted RKHwPRFs from various multilinear maps and associated assumptions. Our framework gives several new results, such as:

- The first iO scheme that relies *only* on SXDH over any asymmetric multilinear map without additional assumptions.
- The first iO scheme that relies *only* on DLIN (or more generally Matrix-DDH) over any (even symmetric) multilinear map without additional assumptions.
- The first iO scheme that relies on SXDH over the multilinear map presented by Ma and Zhandry (TCC'18) (the authors only presented a NIKE protocol in their paper). To our knowledge, this candidate multilinear map has not been successfully cryptanalyzed, and the SXDH assumption plausibly holds over it.

Our analysis of RKHwPRFs in a sense completes the work initiated by Alamati *et al.* (EUROCRYPT'19) on building cryptosystems from generic Minicrypt primitives with structure. With our results, almost all of the major known cryptosystems can be built from a weak PRF with either a group or ring homomorphism over either the input space or the key space. Thus, a major contribution of this work is advancing the study of the relationship between structure and cryptography.

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

1 Introduction

A long-standing line of work in the research of theoretical cryptography has been to build cryptosystems with more functionalities from more structured mathematical assumptions. A major initial development in this direction was the invention of public-key encryption (PKE) [RSA78, DH76]. Many years later, the development of pairing-based cryptography allowed for more exciting constructions such as identity-based encryption [BF01] and three-party noninteractive key exchange [Jou00], both of which were not known from previously studied assumptions. While pairing-based cryptography needed stronger assumptions than previously known PKE schemes, the absence of classical attacks over the past (almost) twenty years since the introduction of these assumptions has seemingly justified their usage [MSS16].

Soon after pairing-based cryptography was developed, Regev introduced a new lattice-based assumption called learning with errors (LWE) in his seminal paper [Reg05]. This enabled realizing even richer cryptographic functionalities. Most notably, Gentry's candidate construction of fully-homomorphic encryption (FHE) [Gen09] brought exciting new possibilities for secure computation on the cloud. Follow-up works [BV11, Bra12, GSW13] introduced new techniques that not only improved this construction, but also allowed us to build other interesting cryptographic functionalities.

In 2013, Garg, Gentry, and Halevi proposed the first candidate multilinear map construction [GGH13a] which gained a lot of attention of the cryptographic community. Soon after, Garg *et al.* [GGH$^+$13b] showed how to build indistinguishability obfuscation (iO) [BGI$^+$01] from multilinear maps. This was particularly notable since obfuscation seemed to be the holy grail in terms of theoretical cryptographic functionality: virtually everything one might want to build in a cryptosystem is possible with iO (and some other mild assumptions): functional encryption [GGH$^+$13b], multi-party noninteractive key exchange [BZ14], and much more [SW14, GGHR14, HSW14, BP15].

All of these applications spurred a huge interest in building graded encodings[1] and iO from more standard assumptions and using more efficient constructions. In a short time, other candidate graded encoding schemes [CLT13, GGH15] and even direct iO constructions [Zim15, AB15] were proposed. Unfortunately, several of these candidate constructions were cryptanalized, starting with the work of Cheon *et al.* [CHL$^+$15] on the cryptanalysis of the multilinear map defined in [CLT13]. More attacks on iO constructions followed [MSZ16, HJ16, CLLT16], breaking all of the original graded encoding schemes.

Naturally, people attempted to improve existing constructions to be immune to these attacks [CLT15, BMSZ15, GMM$^+$16], but many of these improved constructions were also shown to be vulnerable to attacks [MSZ16, CGH17]. As far as we know, there is only one currently unbroken published multilinear map [MZ18] and it is not known to imply iO.[2]

A series of relatively recent works [Lin16, LV16, LT17, AS17] showed how to reduce the degree of multilinearity needed in a multilinear map in order to build iO using novel techniques such as local PRGs. In fact, in [LT17], the authors showed a construction from bilinear maps and 2-blockwise local PRGs, which seemingly paved the way for a secure iO construction. However, Lombardi and Vaikuntanathan [LV17] and Barak *et al.* [BBKK18] independently showed that 2-blockwise local PRGs could never be constructed securely, unfortunately implying that such iO constructions are not secure.

More recent works [Agr19, AM18, AJL$^+$19, JLMS19] have used a number of interesting and novel new techniques, including using perturbation resilient generators.[3] While these constructions provide many new insights, they are still based on less studied assumptions, and we would like to achieve the "holy grail" of constructing iO from standard cryptographic assumptions. Indeed, the fact that more recent, novel iO assumptions such as (most) multilinear maps and 2-blockwise local PRGs have been broken has caused some consternation in the wider cryptographic community about the plausibility of iO. Bishop *et al.* summarized this sentiment in [BKM$^+$19].

In this paper, we study iO in a way that differs from the traditional approach of directly proposing more assumptions and constructions (although we certainly agree that this traditional approach has plenty of merit). While others have related iO to primitives such as functional encryption [AJ15, BV15] and constraint-hiding constrained PRFs [CC17, BKM17], our approach is motivated by questions such as: what sort of mathematical structure is seemingly sufficient

[1]Graded encodings are generalizations of multilinear maps.

[2]Martin Albrecht and Alex Davidson have created a website on attacks related to graded encodings and iO constructions [Alb19]. Although it is not completely up to date, we refer the reader to this website for more information on concrete iO constructions and attacks.

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

for realizing iO? It is our hope that studying the complexity of iO can help us to either build or rule out constructions of iO from certain assumptions. Due to the promise that iO offers in terms of cryptographic applications, we think this is a very worthwhile goal in theoretical cryptography.

1.1 Structure and Cryptography

It has long been assumed by many in the cryptographic community that there is some kind of relationship between mathematical structure and public-key cryptography. Barak [Bar17] has frequently ruminated on this topic. Very recently, this was formalized in a work by Alamati *et al.* [AMPR19] that focused on the relationship between mathematical structure and cryptographic functionality.

In [AMPR19], the authors showed that applying *input* homomorphisms to simple primitives in Minicrypt like weak PRFs allows us to build many primitives in Cryptomania. For instance, a simple group input-homomorphic weak pseudorandom function (IHwPRF) can be used to build most of the cryptosystems that we know how to build from the DDH assumption, and a *ring* input-homomorphic weak PRF, with some restrictions, is equivalent to FHE. The authors do consider structured functions that are equivalent to bilinear/multilinear maps in the form of "ℓ-composable input homomorphic functions," but their definitions are unsatisfactory since they almost mimic multilinear maps.

In a follow-up work [AMP19], Alamati *et al.* considered simple primitives with structured *secrets*. While this work has some novel insights on *key* homomorphisms [NPR99, BLMR13] over simple primitives, it does not consider higher-order applications like multilinear maps or iO. So despite all of the recent work studying mathematical structure in cryptography, there is very little to show on iO and related primitives like multi-party non-interactive key exchange (NIKE). This is despite the fact that, in our opinion, iO is the *most* important primitive for which we need to study the required mathematical structure, because a good formalization of the structure required for iO would hopefully allows us more insights on the security of constructions (or, perhaps more pessimistically, negative results).

So the question remains: can we come up with simple, structured primitives that imply iO, thus providing more insight on the kind of assumptions (or mathematical structure) that are seemingly sufficient to realize iO?

1.2 Our Contributions

We answer this question in the affirmative. We define new, simple primitives called *ring key-homomorphic weak pseudorandom functions* (RKHwPRF) and *ring-embedded homomorphic sysnthesizers* (RHS). Ring-embedded homomorphic synthesizers are, informally, a substantially weaker form of RKHwPRF where an adversary doesn't get to see the input values to the weak PRF (we define these primitives formally later in the paper). We show how to build iO from a generic RHS (even in the presence of some structural restrictions), and also show that any graded encoding or multilinear map that satisfies some basic assumptions can be used to build a (restricted) RHS. Along the way we discover several different interesting observations on the relationship between ring structure and cryptographic primitives. We outline this in the rest of this section. We refer to Figure 1 for a (simplified) overview of our results.

Definitions. We begin the technical content of our paper by defining several new primitives, including *ring key-homomorphic weak PRFs* and *ring homomorphic synthesizers*.

As an example, we informally say that a weak PRF is *ring key-homomorphic* if, for some weak PRF $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ such that both the keyspace $(\mathcal{K}, \oplus, \otimes)$ and the output space $(\mathcal{Y}, \boxplus, \boxtimes)$ are rings with efficiently computable ring operations, the following hold:

$$F(k_1, x) \boxplus F(k_2, x) = F(k_1 \oplus k_2, x),$$

$$F(k_1, x) \boxtimes F(k_2, x) = F(k_1 \otimes k_2, x).$$

We refer to the above definition as a "classic" ring key-homomorphic weak PRF. We can further generalize this definition to cover a number of additional situations. All of our constructions accommodate approximate (or bounded) homomorphisms like in lattice-based assumptions. We can also handle cases (like in graded encodings or asymmetric multilinear maps) where we have "slots" for elements, and the elements must be multiplied in a certain order. In addition, our constructions allow for what we call "partial" RKHwPRFs where we sacrifice determinism and use some randomness in order to ensure security.

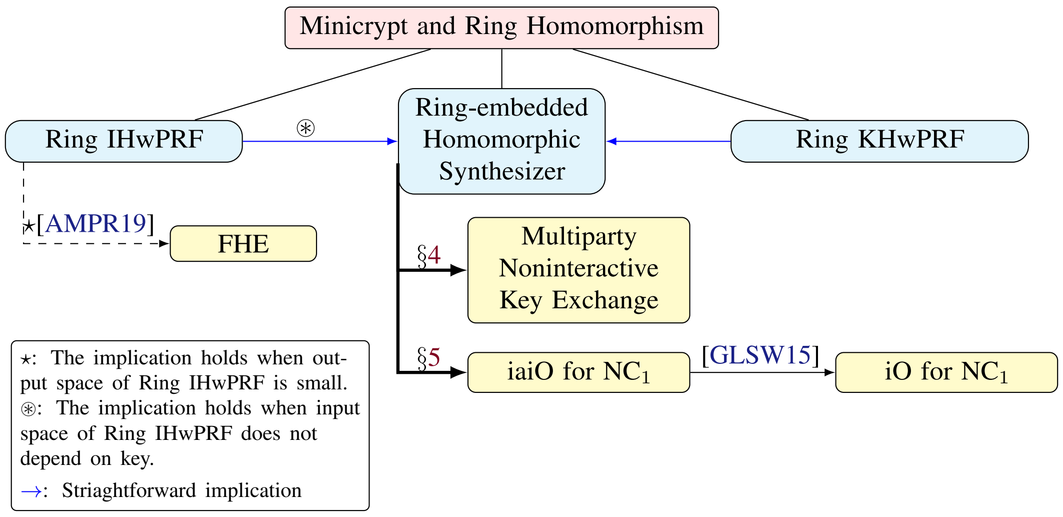
Minicrypt and Ring Homomorphism
Ring IHwPRF
⊛
Ring-embedded Homomorphic Synthesizer
Ring KHwPRF
⋆[AMPR19]
FHE
§4
Multiparty Noninteractive Key Exchange
§5
iaiO for NC$_1$
[GLSW15]
iO for NC$_1$
⋆: The implication holds when output space of Ring IHwPRF is small.
⊛: The implication holds when input space of Ring IHwPRF does not depend on key.
→: Striaghtforward implication

Figure 1: Implications of symmetric primitives with ring homomorphism

Furthermore, our constructions and definitions work for the case where the output space is a *ring embedding*: namely, the output space itself does not have an efficient multiplication algorithm and instead multiplication is carried out by some other algorithm. For instance, bilinear groups are examples of ring embeddings. We fully explain these definitions in more detail later in the paper.

Our definitions can also be extended in the context of pseudorandom synthesizers. Recall that, informally, a pseudorandom synthesizer is a two-input function $S(\cdot,\cdot)$ parameterized by two sets $\mathcal{X}$ and $\mathcal{W}$ such that on random inputs $(x_1, ..., x_m) \in \mathcal{X}^m$ and $(w_1, ..., w_n) \in \mathcal{W}^n$, the matrix $\mathbf{M}$ defined using all mn values of S such that $S(x_i, w_j) = \mathbf{M}_{ij}$ is indistinguishable from random. Synthesizers were introduced by Naor and Reingold in [NR95], where they proved that any secure pseudorandom synthesizer could be used to build a PRF with logarithmic depth in terms of the number of synthesizer evaluations.

We extend these definitions of synthesizers to require that a ring homomorphism hold over *one* of the coordinates of the synthesizer; in other words, for any $x \in \mathcal{X}$ and any $w, w' \in \mathcal{W}$, we have

$$S(x,w) \boxplus S(x,w') = S(x, w \oplus w'),$$

$$S(x,w) \boxtimes S(x,w') = S(x, w \otimes w').$$

Our definition of ring-embedded homomorphic synthesizer (RHS) weakens these requirements even further by requiring that the homomorphism holds *only* with respect to addition, and that multiplication can be efficiently done *only* on the output space of the synthesizer; in other words, we do not actually require the multiplicative homomorphism. It is straightforward to see that a ring-embedded homomorphic synthesizer is implied by any ring-homomorphic synthesizer as defined above, which in turn is implied by any ring key-homomorphic weak PRF (RKHwPRF). We can further generalize RHSs in the same ways we generalize RKHwPRFs. We refer the reader to Section 3 for the formal definitions.

Multi-Party NIKE from RHS. Our first result is a construction of multi-party noninteractive key exchange from a ring-embedded homomorphic synthesizer. As we have suggested already, our construction also immediately implies a construction from a RKHwPRF. We elaborate more on this result in our technical overview, Section 2.

Building iO from RHS. Our second result is a construction of iO (for $\mathcal{NC}^1$) from a ring-embedded homomorphic synthesizer. More concretely, we show how to build an *input-activated* iO from any RHS, which in turn implies standard iO due to [GLSW15]. Our construction is inspired by the [GLSW15] construction. To prove security, we only rely on the security of the RHS and do not need any other assumptions, except if we want to extend past $\mathcal{NC}^1$ functions; in the latter case, we also need FHE [GGH+13b].

Our iO construction is built from a program-independent assumption (the security of the RHS) and is in the standard model.[1] We do not restrict the adversary to any specific kind of attacks.

We develop many new techniques for dealing with noncommutative rings in our construction and proof. Our techniques depart significantly from those used in existing iO constructions based on multilinear maps/graded encodings, since these constructions are typically based on hardness assumptions over commutative rings. We believe that these techniques are of independent interest and would hopefully enable building other cryptoprimitives over generic rings.

iO from SXDH on Multilinear Maps. Gentry *et al.* show that a generic multilinear map that satisfies the multilinear subgroup elimination assumption implies iO in their seminal work [GLSW15].

In our work, we show that a generic multilinear map that satisfies the SXDH assumption implies a "slotted" RKHwPRF, which implies an iO construction. Thus, we show that multilinear maps where SXDH holds can be used to build iO, generalizing the result of [GLSW15] to a wider class of multilinear maps. To our knowledge, previous work that built iO from multilinear maps assuming SXDH [Lin17, LT17] required other assumptions, such as blockwise-local PRGs and subexponentially secure LWE.

iO from Matrix DDH on Multilinear Maps. Most iO constructions rely on asymmetric multilinear maps that inherently impose certain restrictions on the ability to pair elements from different groups. In this work, we propose new techniques to build iO from relatively mild assumptions over any multilinear map (including symmetric ones).

More concretely, we show that any multilinear map where DLIN (and more generally the matrix DDH family of assumptions [EHK+13]) holds can be used to build a slotted partial RKHwPRF. To our knowledge, this implies the first construction of iO from any multilinear map assuming only DLIN (or matrix-DDH) without any additional assumptions.

iO from "The MMap Strikes Back" Multilinear Map. In [MZ18], Ma and Zhandry propose a candidate multilinear map construction based on repeated instances of the [CLT13] multilinear map construction. To our knowledge, this paper contains the only published multilinear map which has not been successfully cryptanalyzed.[2] However, the construction is substantially less structured than previous graded encodings, and Ma and Zhandry are only able to construct NIKE using their multilinear map–they leave iO as an open problem.

In this work, we show how to build a slotted "partial" RKHwPRF based on the multilinear map construction in "The MMap Strikes Back." Assuming SXDH holds over this multilinear map (which, as far as we know, is unbroken so far), we can construct a secure slotted partial RKHwPRF, which is sufficient to realize a secure iO construction. To our knowledge, this would be the only non-cryptanalyzed construction of iO from a multilinear map.

We think that this construction is an excellent example of the power of our techniques: the [MZ18] construction is very unstructured, and previously existing methods for building iO from graded encodings require much more structure than this construction has. However, the unstructured nature of (slotted, partial) RKHwPRFs allows us show that such a graded encoding does, in fact, imply a (slotted, partical) RKHwPRF, which means we can get an iO construction.

Multilinear Map, iO, and RKHwPRF Equivalence. We have already discussed a multitude of papers that build iO from multilinear maps. However, a recent line of work [AFH+16, FHHL18] has focused on the converse: namely, showing how to build multilinear maps from iO. In particular, [FHHL18] constructs a multilinear map where the *multilinear* DDH (MLDDH) assumption holds from iO.

In a more recent work, Alamati *et al.* [AMP20] showed how to build multilinear maps endowed with most of the well-known (prime order) "source group" assumptions from subexponentially secure iO and some other (relatively) standard assumptions. In particular, they showed how to build an SXDH-hard multilinear map with randomized encodings. Since we show in this work that such a multilinear map implies a slotted partial RKHwPRF, it follows that, assuming subexponential security and standard assumptions, a slotted partial RKHwPRF can be built from iO itself.

[1]As discussed in [GLSW15], the reduction from a program-independent assumption seems to inherently imply an exponential security loss.

[2]We note that there are maps like [BGMZ18] which have been cryptanalyzed [CCH+19] but, to our knowledge, not necessarily broken in all possible parameter regimes.

This implies that, modulo subexponential security and standard assumptions, (slotted partial) RKHwPRFs are in a sense "iO-complete", and can be built from other primitives known to imply iO, e.g., compact functional encryption [AJ15, BV15] and, from a more recent work, split FHE [BDGM20]. Thus, RKHwPRFs both imply and are implied by a very wide range of (subexponentially secure) powerful primitives (note that subexponential security a common requirement for reductions in this space, e.g., constructions of iO from compact FE [AJ15, BV15]).

Field-Homomorphic Synthesizers Are Impossible. Given the implication that a RHS is sufficient to realize iO, it is natural to ask whether it is possible to have a stronger version of a RHS where the output space is a field with efficiently computable field operations (we call such a primitive a field-embedded homomorphic synthesizer, or FHS in short). We answer this question in negative by showing that there is no secure FHS. Since an FHS is implied by a field key-homomorphic weak PRF (FKHwPRF), it follows that there is no secure FKHwPRF as well.

Previously, Maurer and Raub [MR07] showed that field-homomorphic one-way permutations were impossible, and our work extends this result. Moreover, it seems unlikely that our attacks here can be extended to the ring case. In particular, it is not known how to compute kernels or inverses over general rings, which makes our attack on fields infeasible to trivially extend to rings. We refer the reader to [ADM06, Jag12, YYHK18] for discussions on the hardness of computing inverses/kernels over generic rings and its implications.

It is easy to see that this negative result does not apply to multilinear maps since the number of multiplication operations over the output space is apriori bounded by the degree of multilinearity and inverses are infeasible to compute over the output space.

New Separation Results. Our findings have interesting implications in terms of separation results. Garg *et al.* [GMM17] showed that certain powerful "all-or-nothing" primitives, e.g., witness encryption and FHE, cannot be used in a black-box manner to construct iO. In this work, we show that ring-embedded homomorphic synthesizers (and ring key-homomorphic weak PRFs) imply iO in a black-box manner, thereby ruling out black-box constructions of these generic primitives with structure from witness encryption and FHE.

As a side note, in [AMPR19] the authors show that FHE implies ring *input*-homomorphic weak PRFs (RIHwPRFs) with certain restrictions. Our results thus also rule out the possibility of building RHS (and RKHwPRFs) in a black-box manner from such restricted RIHwPRFs.

Outline. The rest of the paper is organized as follows. Section 2 provides an overview of our techniques. Section 3 provides preliminary background material. Section 4 presents the construction and proof of multiparty NIKE from RHS. Section 5 presents the construction of iaiO for $\mathcal{NC}^1$ from RHS. Section 6 introduces our subspace hiding assumption, and Section 7 proves the security of our iaiO construction based on this assumption. Sections 8 and 9 prove the subspace hiding assumption. Section 13 rules out the existence of field-homomorphic synthesizers. Sections 11 and 13 formally define slotted RKHwPRFs and slotted partial RKHwPRFs, and show how to instantiate them from various multilinear maps and associated assumptions. Finally, Section 14 presents conclusing remarks and open questions.

2 Technical Overview

In this section, we explain the intuition behind our constructions and proofs at a high level. We refer to the remainder of the paper (where the formal definitions and proofs are located) as appropriate. We will generally write all of our results in this section in terms of ring key-homomorphic weak PRFs (RKHwPRFs) in order to simplify our arguments, although our results will also follow from a substantially weaker primitive, namely ring-embedded homomorphic synthesizers.

2.1 Definitions and Intuition

Recall that we informally say that a weak PRF is *ring key-homomorphic* if, for some weak PRF $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ with both keyspace $(\mathcal{K}, \oplus, \otimes)$ and output space $(\mathcal{Y}, \boxplus, \boxtimes)$ rings with efficiently computable ring operations, the following hold:

$$F(k_1, x) \boxplus F(k_2, x) = F(k_1 \oplus k_2, x),$$

$$F(k_1, x) \boxtimes F(k_2, x) = F(k_1 \otimes k_2, x).$$

We call this a *classic* RKHwPRF. We write our proofs and constructions using these classic RKHwPRFs. However, in many cases, some of the underlying objects (say, graded encodings or multilinear maps) for which we want to construct iO or NIKE do not quite fit into the framework of a classic RKHwPRF. To accomodate this, we consider more generic versions of the definition of an RKHwPRF. We note that all of the proofs and constructions work in essentially the same way for all versions of the definitions we provide here.

Slotted RKHwPRFs. Suppose we consider an asymmetric bilinear map where the symmetric external Diffie-Hellman (SXDH) assumption holds. Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be an asymmetric efficiently computable non-degenerate bilinear map with "source groups" $\mathbb{G}_1$ and $\mathbb{G}_2$, and target group $\mathbb{G}_T$, where each group is order q (assumed prime). Also, let $\mathcal{X}$ be a set of size q such that there exist efficiently computable "encoding functions":

$$H_1 : \mathcal{X} \to \mathbb{G}_1, \quad H_2 : \mathcal{X} \to \mathbb{G}_2, \quad H_T : \mathcal{X} \to \mathbb{G}_T,$$

such that for any $x \in \mathcal{X}$, we have

$$H_T(x) = e(H_1(x), H_2(x)).$$

Note that we can create such functions by setting H_1 and H_2 to be efficiently computable bijections, and letting H_T be the bilinear map evaluation of their output.

Now, for $i \in \{1, 2, T\}$, define the function $F_i : \mathbb{Z}_q \times \mathcal{X} \to \mathbb{G}_i$ as:

$$F_i(k, x) = H_i(x)^k.$$

Note that assuming SXDH holds (implying that DDH is hard over each individual source group $\mathbb{G}_1$ and $\mathbb{G}_2$, and hence, over the target group $\mathbb{G}_T$), we can state the following:

- If the "encoding functions" H_1 and H_2 are modeled as *bijections*, each F_i for $i \in \{1, 2, T\}$ is a weak PRF.
- Each F_i for $i \in \{1, 2, T\}$ is homomorphic with respect to addition. More concretely, for any $k_1, k_2 \in \mathbb{Z}_q$ and for any $x \in \mathcal{X}$, we have:
$$F_i(k_1 + k_2, x) = F_i(k_1, x) \cdot F_i(k_2, x).$$

This gives us a set of related group key-homomorphic weak PRFs. However, defining ring-homomorphism is tricky, since none of $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ are necessarily equipped with efficiently computable multiplication operations (in fact, for some of these groups, such an operation may not even be properly defined).

However, note that we can use the bilinear map e to "simulate" a single multiplication operation as follows: for any $k_1, k_2 \in \mathbb{Z}_q$ and for any $x \in \mathcal{X}$, define the following operation:

$$F_1(k_1, x) \odot F_2(k_2, x) := e(F_1(k_1, x), F_2(k_2, x)).$$

It is easy to see that $F_1(k_1, x) \odot F_2(k_2, x) = F_T(k_1 \cdot k_2, x)$. While such a construction does not formally meet the definition of a "classic" RKHwPRF, it does give us something that seemingly provides the same functionality of an RKHwPRF. We can thus view the ensemble $\{F_1, F_2, F_T\}$ as a 2-"slotted"-RKHwPRF, that supports unbounded addition operations and a single multiplication operation, with the added restriction that a multiplication is only allowed between elements from $\mathbb{G}_1$ (which constitutes "slot-1" of the output space) and $\mathbb{G}_2$ (which constitutes "slot-2" of the output space), while no multiplications involving elements from $\mathbb{G}_T$ (which constitutes "slot-2" of the output space) are allowed.

Note that for the multiplicative homomorphism to hold in "slotted" RKHwPRFs, we need to ensure that we can provide outputs on the same inputs across all slots (where each "slot" has a different key); in other words, a "slotted" RKHwPRF resembles a collection of parallel-secure weak PRFs. We formalize slotted RKHsPRFs using notion of "interval encodings" [MZ18]. In a nutshell, slotted RKHwPRFs are essentially classic RKHwPRFs with the following restrictions:

1. The elements in the output space are divided into n singleton "slots" or intervals $[i]$ for $i \in [1, ..., n]$.

2. The output space of any PRF in this family is not equipped with an efficient multiplication algorithm by itself. Instead multiplication is "simulated" by using the graded encoding operation.

3. Elements in adjacent intervals $[i, j]$, $[j + 1, k]$ can be multiplied, getting a new element in the interval $[i, k]$. This requires the use of a graded encoding multiplication operation.

4. Two elements in the same interval can be added together (as usual), but elements not in the same interval cannot be added.

5. Unlike the standard notion of RKHwPRFs where one can perform unbounded many multiplication operations in the output space, the number of allowed multiplications in the output space of n-"slotted"-RKHwPRFs is upper bounded by $(n - 1)$, where n is the degree of multilinearity of the underlying asymmetric multilinear map. Note that the number of allowed additive operations is still unbounded.

We note that, like most other iO constructions using high-degree multilinear maps, we require that we can incrementally evaluate multiplications on the multilinear maps, so our constructions would not work for any (hypothetical) multilinear maps that only allow "one-shot" pairings of all n elements together to a final element in the target group. Notably, some constructions using low-degree multilinear maps and working through functional encryption [Lin17, AS17] do not have this restriction and only require "one-shot" pairings.

Applications of Slotted RKHwPRFs. It turns that all of the constructions presented in this paper can be realized not only from RKHwPRFs but also from n-"slotted"-RKHwPRFs, where the choice of parameter n would depend on the corresponding construction. For example, in the construction of multiparty key-exchange, it suffices to set $n = N - 1$, where N is the number of involved parties. Similarly, for our iO construction, it suffices to set $n = L + N - 1$, where L is the depth of the permutation branching program corresponding to the circuit to be obfuscated and N is the number of input variables.

Correctness and security of our RKHwPRF constructions hold immediately for slotted RKHwPRFs for (mainly) one reason: the multiplications of ring elements in all of our constructions are done in a fixed order that is independent of the input and set in advance. Fundamentally, for our constructions a slotted RKHwPRF is no different than a standard one, but the more general construction lets us capture more instantiations.

In Section 11, we show that any asymmetric multilinear map of degree n where the SXDH assumption holds implies a family of n-"slotted"-RKHwPRFs. Coupled with our iO construction from RKHwPRFs, this allows us to achieve iO from a multilinear map where SXDH holds. The analysis is very similar to what we have discussed above for bilinear maps.

Slotted Partial RKHwPRFs. In Section 12, we consider a variation of slotted RKHwPRFs to take into account multilinear maps with "noisy" encodings [CLT13, MZ18]. We call this variation slotted "partial" RKHwPRFs, and show how to instantiate it from any multilinear map with noisy encodings where the SXDH assumption holds. Our definitions of slotted partial RKHwPRFs are similar in flavor to the definitions of "almost" key-homomorphic PRFs in [BLMR13], albeit with an additional zero test.[1] All of our constructions also follow from slotted "partial" RKHwPRFs.

In Section 12, we also show how to build slotted partial RKHwPRFs from either of the following assumptions: (a) the SXDH assumption over the asymmetric multilinear map of Ma and Zhandry [MZ18], and (b) DLIN (or more generally, matrix-DDH) over any (possibly symmetric) multilinear map. Below, we present some of the ideas behind the second construction.

Let $e : \mathbb{G} \times \ldots \times \mathbb{G} \to \mathbb{G}_T$ be a symmetric N-linear map such that the matrix DDH assumption (with appropriate dimension parameter $m > N$) holds over the group $\mathbb{G}$. To begin with, observe that the function $F : \mathbb{Z}_q^{m\times m} \times \mathbb{G}^{m\times m} \to \mathbb{G}^{m\times m}$ defined as

$$F(\mathbf{K}, g^{\mathbf{X}}) = g^{\mathbf{XK}},$$

is a weak PRF. This weak PRF is additively key-homomorphic, but the non-commutativity of matrix multiplication presents an obstacle to achieving multiplicative key-homomorphism via the pairing operation.

[1]Note that evaluating a classical/slotted RKHwPRF on any input using the key $k = 0_{\mathcal{K}}$ trivially results in a zero output, which can be used as a zero test. Since this is not the case when the homomorphism is partial, we need an explicit zero test.

In order to address this, we resort to designing a slotted weak PRF family with additional correlated randomness between "adjacent" interval-slots. At a high level, for a given input x, a weak PRF in the slot $[i, j]$ for $i \leq j \leq 2$ has the following form:

$$F_{i,j}((K, \phi_{i,(j+1)}), x) = g^{\mathbf{A}_{(x,i)}\mathbf{K}\mathbf{A}^{-1}_{(x,(j+1))}},$$

where $\phi_{i,(j+1)}$ is a specially structured random secret used to generate the random matrices $\mathbf{A}_{x,i}$ and $\mathbf{A}_{x,j+1}$ in $\mathbb{Z}_q^{m\times m}$. We refer the reader to Section 12 for the proof of weak pseudorandomness based on the matrix DDH assumption.

Note that in this formulation, we can exploit the common random terms between "adjacent" slots-pairs $([i, j], [j + 1, \ell])$ to achieve multiplicative homomorphism, albeit over "part" of the secret key. Nonetheless, as we explain in Section 12, such a slotted "partial" RKHwPRF suffices for our constructions.

2.2 Key Homomorphism and Subset Sum

The starting point of our work is the main technique from [AMP19], where the authors observe that "repeated" random subset sums (i.e., subset sums computed over different sets of elements using the same random subset) over the output space of a KHwPRF are indistinguishable from uniformly random elements. This may sound counterintuitive, but follows from a relatively simple observation. Let $F := \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ be a KHwPRF where $(\mathcal{K}, \oplus)$ and $(\mathcal{Y}, \boxplus)$ are groups with efficient group operations. Suppose $\mathbf{b}$ is a random binary string of length n. Now consider the following sum over the set of KHwPRF outputs, for randomly chosen keys k_i and a random inputs x_j, where if $\mathbf{b}_i$ is one we include the ith row in our sum, and if $\mathbf{b}_i$ is zero, we do not:

$$\begin{array}{rlcccccl}
 & \mathbf{b}_1\,[& F(k_1, x_1) & F(k_1, x_2) & F(k_1, x_3) & \dots & F(k_1, x_m) &] \\
 & \mathbf{b}_2\,[& F(k_2, x_1) & F(k_2, x_2) & F(k_2, x_3) & \dots & F(k_2, x_m) &] \\
 & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\
\boxplus & \mathbf{b}_n\,[& F(k_n, x_1) & F(k_n, x_2) & F(k_n, x_3) & \dots & F(k_n, x_m) &] \\
\hline
 & & F(k', x_1) & F(k', x_2) & F(k', x_3) & \dots & F(k', x_m) &
\end{array}$$

Note that if $n > 3\log|\mathcal{K}|$, then k' is distributed *statistically* close to random by the leftover hash lemma [IZ89]. This means that the resulting sums–which are valid PRF outputs–are *computationally* indistinguishable from random even if $m >> n$ due to the security of the KHwPRF; in other words we can repeatedly subset sum using the same random subset arbitrary (polynomially) many times and still produce pseudorandom group elements–assuming the binary vector $\mathbf{b}$ is hidden, the whole ensemble of PRF outputs above will be indistinguishable from random. In addition, we note that, since KHwPRF outputs are indistinguishable from random in the output space $\mathcal{Y}$, the above arguments apply not just to KHwPRF outputs but also randomly sampled elements in $\mathcal{Y}$ as well.

In [AMP19], the authors use this technique to show that KHwPRFs imply PKE and other asymmetric primitives in Cryptomania, but that is the extent of their application. It turns out that if we generalize to RKHwPRFs, we can considerably increase the power of this technique.

2.3 Ring Homomorphism and Subset Sum

Now suppose we modify our weak PRF F to be an RKHwPRF, with the keyspace $(\mathcal{K}, \oplus, \otimes)$ and output space $(\mathcal{Y}, \boxplus, \boxtimes)$ having efficiently computable ring operations. Suppose that we consider a matrix of wPRF outputs

$$\mathbf{K}_x = \begin{array}{ccccc}
F(k_{1,1}, x) & F(k_{1,2}, x) & F(k_{1,3}, x) & \dots & F(k_{1,n}, x) \\
F(k_{2,1}, x) & F(k_{2,2}, x) & F(k_{2,3}, x) & \dots & F(k_{2,n}, x) \\
\dots & \dots & \dots & \dots & \dots \\
F(k_{n,1}, x) & F(k_{n,2}, x) & F(k_{n,3}, x) & \dots & F(k_{n,n}, x)
\end{array}$$

where we have made $\mathbf{K}_x$ square. If we have some other matrix $\mathbf{S}_x \in \mathcal{Y}^{n\times n}$ sampled in exactly the same way, then, by the ring key-homomorphism, the product $\mathbf{S}_x\mathbf{K}_x$ is well-behaved: the (i, j)th entry of $\mathbf{S}_x\mathbf{K}_x$ is just $F(\tilde{k}, x)$, where $\tilde{k}$ is the (i, j)th entry of the products of the matrices of the corresponding keys of $\mathbf{S}_x$ and $\mathbf{K}_x$ respectively.

We now illustrate a consequence of this observation. Suppose that we sample matrices $\mathbf{S}_x^1, ..., \mathbf{S}_x^m \in \mathcal{Y}^{n\times n}$ with *independent keys* as described above. Then the tuple

$$\left(\begin{bmatrix} \mathbf{S}_x^1 \\ \mathbf{S}_x^2 \\ ... \\ \mathbf{S}_x^m \end{bmatrix}, \begin{bmatrix} \mathbf{S}_x^1 \\ \mathbf{S}_x^2 \\ ... \\ \mathbf{S}_x^m \end{bmatrix} [\mathbf{K}_x] \right)$$

is indistinguishable from random. This follows from a similar argument that was used to show that subset sums with reused randomness are hard for KHwPRFs. We develop new techniques to be able to move from repeated subset sums over groups to matrix multiplications over rings while retaining the pseudorandomness of the final elements. Concretely, the intuition behind our proof techniques is as follows:

Step 1: Apply the leftover hash lemma to argue that the dot product of two random vectors over the keyspace of an RKHwPRF results in a key that is statistically indistinguishable from random. This step is conceptually similar to the first step in the previous proof, where we used the leftover hash lemma to argue that a single subset sum over a vector of KHwPRF keys using a random subset results in a key that is statistically indistinguishable from random. In the ring setting, we esentially shift from using subset sums to using general ring-linear sums, while still retaining the ability to invoke the leftover hash lemma when required.

Step 2: Invoke the security of the RKHwPRF to expand from vector dot products to matrix products, while retaining *computational* indistinguishability from random. Again, this step is conceptually similar to the second step in the previous proof, where we invoked the security of the KHwPRF to go from a single subset sum to repeated subset sums, while retaining *computational* indistinguishability from random.

Once again, due to the fact that our wPRF outputs must be indistinguishable from random, we can work with random elements in the output ring $\mathcal{Y}$ rather than outputs of the RKHwPRF itself, and the output will still be indistinguishable from random. In other words, we prove the following claim: for random matrices $\mathbf{R} \in \mathcal{Y}^{m\times n}$ and $\mathbf{T} \in \mathcal{Y}^{n\times k}$, the tuple $(\mathbf{R}, \mathbf{RT})$ is indistinguishable from random even if $m >> n$ and $k >> n$. This turns out to be a very powerful technical tool that we use extensively in all our constructions, including those based on the significantly weaker primitive, namely ring-embedded homomorphic synthesizer.

Our techniques extend naturally in the context of "slotted" RKHwPRFs, even though the output space for the same is not an explicit ring. We essentially simulate multiplication operations over the output space of a "slotted" RKHwPRF using machinery provided by multilinear maps. Note that the key space for a "slotted" RKHwPRF is still a ring, which allows us to invoke the leftover hash lemma for ring-linear sums over this space. Finally, we can invoke the weak pseudorandomness properties of the RKHwPRF to argue that repeated wPRF evaluations using the same ring-linear sum over the key space results in elements that are computationally indistinguishable from uniform.

2.4 Multiparty Noninteractive Key Exchange

Our first construction involves building noninteractive key exchange from RHS (in this overview, we will assume an RKHwPRF for ease of exposition). This construction is relatively simple and relies on our new technique to show the hardness of distinguishing simple matrix products over RKHwPRF output spaces from random. In this overview, we will focus on the 3-party case instead of the N-party case for simplicity. The intuition for the N-party case follows similarly.

Given an RKHwPRF $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$, we first fix parameters $m > 3\log|\mathcal{K}|$ and $n > 6m^2\log(|\mathcal{Y}|)$. Let $\mathcal{R} = M_m(\mathcal{Y})$ denote the ring of all m by m square matrices over $\mathcal{Y}$. We remark that $\log(|\mathcal{R}^{n\times n}|)$ is polynomial in the security parameter, and hence elements of $\mathcal{R}^{n\times n}$ can be represented using polynomially many bits.

Our public parameters for 3-party NIKE are going to be two matrices $\mathbf{R}^{(1)}$ and $\mathbf{R}^{(2)}$ sampled uniformly at random from $\mathcal{R}^{n\times n}$, where $\mathcal{R}$ is the matrix ring as defined above. Our proposed protocol works as follows:

Alice	Bob	Charlie
Sample $\mathbf{S}_A \leftarrow \mathcal{R}^{n\times n}$	Sample $\mathbf{S}_B \leftarrow \mathcal{R}^{n\times n}$	Sample $\mathbf{S}_C \leftarrow \mathcal{R}^{n\times n}$
Publish $\mathbf{P}_A = \mathbf{S}_A\mathbf{R}^{(1)}$	Publish $\mathbf{P}_B^{(1)} = \mathbf{R}^{(1)}\mathbf{S}_B$	Publish $\mathbf{P}_C = \mathbf{R}^{(2)}\mathbf{S}_C$
	$\mathbf{P}_B^{(2)} = \mathbf{S}_B\mathbf{R}^{(2)}$	

The final shared secret is $\boxed{\mathbf{S}} := \mathbf{S}_A\mathbf{R}^{(1)}\mathbf{S}_B\mathbf{R}^{(2)}\mathbf{S}_C$. Alice/Bob/Charlie can compute the final secret $\boxed{\mathbf{S}}$ as

$$\begin{aligned}\boxed{\mathbf{S}} = \mathbf{S}_A\mathbf{R}^{(1)}\mathbf{S}_B\mathbf{R}^{(2)}\mathbf{S}_C &= \mathbf{S}_A\mathbf{P}_B^{(1)}\mathbf{P}_C && \text{(Alice)}\\ &= \mathbf{P}_A\mathbf{S}_B\mathbf{P}_C && \text{(Bob)}\\ &= \mathbf{P}_A\mathbf{P}_B^{(2)}\mathbf{S}_C && \text{(Charlie).}\end{aligned}$$

While the construction is relatively simple, the security proof is more involved (in particular, when generalizing to an arbitrary number of parties). We need to show that the following tuples are indistinguishable from each other:

$$\begin{gathered}\left(\mathbf{R}^{(1)}, \mathbf{R}^{(2)}, \mathbf{S}_A\mathbf{R}^{(1)}, \mathbf{R}^{(1)}\mathbf{S}_B, \mathbf{S}_B\mathbf{R}^{(2)}, \mathbf{R}^{(2)}\mathbf{S}_C, \mathbf{S}_A\mathbf{R}^{(1)}\mathbf{S}_B\mathbf{R}^{(2)}\mathbf{S}_C\right),\\ \left(\mathbf{R}^{(1)}, \mathbf{R}^{(2)}, \mathbf{S}_A\mathbf{R}^{(1)}, \mathbf{R}^{(1)}\mathbf{S}_B, \mathbf{S}_B\mathbf{R}^{(2)}, \mathbf{R}^{(2)}\mathbf{S}_C, \mathbf{U}\right),\end{gathered}$$

where $\mathbf{U}$ is a uniformly sampled matrix in $\mathcal{R}^{n\times n}$.

It follows from the assumption that F is an RKHwPRF that the tuples $\left(\mathbf{R}^{(1)}, \mathbf{S}_A\mathbf{R}^{(1)}\right)$ and $\left(\mathbf{R}^{(1)}, \mathbf{T}_A\right)$ are indistinguishable for some random matrix $\mathbf{T}_A$ in $\mathcal{R}^{n\times n}$. We can apply a similar line of argument to the term containing $\mathbf{S}_C$ as well. Thus, we can reduce our above assumption to distinguishing between the following tuples:

$$\begin{gathered}\left(\mathbf{R}^{(1)}, \mathbf{R}^{(2)}, \mathbf{T}_A, \mathbf{R}^{(1)}\mathbf{S}_B, \mathbf{S}_B\mathbf{R}^{(2)}, \mathbf{T}_C, \mathbf{T}_A\mathbf{S}_B\mathbf{T}_C\right),\\ \left(\mathbf{R}^{(1)}, \mathbf{R}^{(2)}, \mathbf{T}_A, \mathbf{R}^{(1)}\mathbf{S}_B, \mathbf{S}_B\mathbf{R}^{(2)}, \mathbf{T}_C, \mathbf{U}\right).\end{gathered}$$

The difficult step in the proof involves implicitly showing that giving an adversary both $\mathbf{R}^{(1)}\mathbf{S}_B$ and $\mathbf{S}_B\mathbf{R}^{(2)}$ does not allow the adversary to learn "enough" about $\mathbf{S}_B$ to distinguish the final term from random.

To do this, we exploit the fact that any uniformly random matrix (with large enough dimensions) in the output ring of the KHwPRF is computationally indistinguishable from a tensor product of two uniformly random vectors in the output ring of the KHwPRF. We introduce and prove certain statistical lemmas with respect to modules that, when combined with the aforementioned observation, allow us to argue that the secret matrix $\mathbf{S}_B$ is computationally hidden, even given both $\mathbf{R}^{(1)}\mathbf{S}_B$ and $\mathbf{S}_B\mathbf{R}^{(2)}$. The security of the overall protocol then follows from this argument. We refer the reader to Section 4 for the detailed proof.

2.5 Indistinguishability Obfuscation

Our second (and main) contribution is a construction of iO for $\mathcal{NC}^1$ from any RKHwPRF. In Section 5, we in fact present an iO construction from a substantially weaker primitive, namely ring-embedded homomorphic synthesizer (RHS). However, for simplicity of exposition, we present the construction from RKHwPRFs here.

We use the framework of [GLSW15] in order to prove iO security, as this is one of the few constructions builds iO from a program-independent assumption and in the standard model.[1] More precisely, we build an input-activated iO scheme and adopt the machinery from [GLSW15] which shows that an input-activated iO scheme implies a full iO scheme.

In [GLSW15], the authors present an input-activated iO construction for $\mathcal{NC}^1$ from composite order multilinear maps. More specifically, they base the security of their construction on an assumption called the *multilinear subgroup elimination assumption*. While their construction introduces many novel ideas for proving the security of iO from a program-independent assumption, their techniques are heavily tuned towards abelian groups. One of the key challenges that we faced when trying to port their ideas to the setting of a generic primitive such as RKHwPRF is the lack of similar useful properties, such as commutative multiplication and cyclic subgroups with explicit representation.[2] Therefore, even though at a very high level, our input-activated iO construction has certain similarities with that of [GLSW15], we use substantially different techniques to achieve both correctness and security.

[1]Most of the other program independent, standard-model constructions build iO through functional encryption [Lin17, LT17].

[2]As the authors of [AMP19] pointed out, such explicit representations are not available in the output space of RKHwPRFs; otherwise an adversary can break the security of the primitive by solving a system of modular equations.

We construct input-activated iO from generic RKHwPRFs as follows: we introduce an assumption (based on a generic primitive, namely RKHwPRF) that can be loosely described as a nonabelian, matrix analogy of the multilinear subgroup elimination assumption. We prove that this assumption holds over the output ring R of any RKHwPRF. We then show that such an assumption implies an input-activated iO scheme.

Construction Ideas and Challenges. Assume that any $\mathcal{NC}_1$ program is represented as an oblivious permutation branching program P of width 5. Let $P = \{M_{\ell,0}, M_{\ell,1}\}_{\ell\in[L]}$, where L denotes the depth of P and each $M_{\ell,b} \in \{0,1\}^{5\times 5}$ is a permutation matrix. The first step in our construction is to embed each program matrix $M_{\ell,b}$ into a matrix $\mathbf{M}_{\ell,b}$ of dimension m by m over the output ring R of the KHwPRF. We refer the reader to Section 5 for the details of the embedding technique.

The next step is to randomize each ring-embedded matrix $\mathbf{M}_{\ell,b}$ using a Kilian-style randomization technique [Kil88]. A natural approach (in line with existing constructions [GGH$^+$13b]) is to generate a sequence of uniformly random matrices $\{\mathbf{Y}_\ell\}_{\ell\in[L]} \leftarrow R^{m\times m}$ and to use the a sequence of encodings $\{\mathbf{N}_{\ell,b}\}_{\ell\in[L],b\in\{0,1\}}$, where

$$\mathbf{N}_{\ell,b} = \mathbf{Y}_\ell^{-1}\mathbf{M}_{\ell,b}\mathbf{Y}_{\ell+1},$$

where $\mathbf{Y}_{L+1}$ is assumed to be the identity matrix $\mathbf{I}$. This would preserve the program's functional behavior.

However, recall that we are working over the output ring R of an RKHwPRF. This essentially means that there is no efficient algorithm to compute inverses in R, (or they might not even exist for most elements). Even worse, computing matrix pseudoinverses over the ring R is hard, since an efficient algorithm for the same would translate into an attack on the security of the RKHwPRF.

Some existing constructions based on generic graded encodings [BR14b] get around this issue by using the adjoint matrix $\mathbf{Z}_\ell = \mathsf{adj}(\mathbf{Y}_\ell)$ instead of the inverse matrix $\mathbf{Y}_\ell^{-1}$, where $\mathbf{Z}_\ell$ is composed of determinants of minors of $\mathbf{Y}_\ell$. However, these constructions use matrix encodings of constant dimensions, for which the adjoint matrices are efficiently computable.

It turns out that in our construction, we must use large matrices (i.e., of dimension $3\log|R|$) in order to be able to use the leftover hash lemma. So it does not seem likely that the novel techniques of [BR14b] can be applied to our construction. Therefore, for our iO construction, we have to develop an entirely new toolbox that deals with matrices of ring elements.

A Simplified Version of Our Construction. We now show a step-by-step process for building up a simplified version of our iO construction. In particular, we show how to build an input-activated iO scheme. Using [GLSW15], we can leverage this into a full iO scheme for all functions in $\mathcal{NC}^1$. We define all of the relevant definitions and notions of security in Section 3, while the detailed construction and proof of security are presented in Section 5 and Section 7. Below, we describe a simplified version of our construction that captures many of the ideas that the full construction builds on.

1. **Permutation Branching Programs.** We assume that any $\mathcal{NC}^1$ program is represented as an oblivious permutation branching program P of width 5. Let $P = \{M_{\ell,0}, M_{\ell,1}\}_{\ell\in[L]}$, where L denotes the depth of P and each $M_{\ell,b} \in \{0,1\}^{5\times 5}$ is a permutation matrix. Also, let $x_1, \ldots, x_N$ denote the input variables, and let $\phi : [L] \to [N]$ be a mapping such that for each $\ell \in [L]$, $i = \phi(\ell)$ indicates which variable x_i controls the ℓ^{th} level branch.

2. **Killian-Style Randomization.** We uniformly sample *additional permutation matrices* $Z_1, \ldots, Z_{L-1} \in \{0,1\}^{5\times 5}$, and define a new set of matrices $\{N_{\ell,b}\}_{\ell\in[L],b\in\{0,1\}}$, where $N_{\ell,b} = Z_{\ell-1}^{-1}M_{\ell,b}Z_\ell$, where Z_0 and Z_L are both set to be the identity permutation. Note that the set of all permutation matrices over $\{0,1\}^{5\times 5}$ form a group, which implies that the aforementioned randomization technique information-theoretically hides the original set of permutation matrices, while retaining the program behavior as is.

3. **Ring-Embedding Permutation Matrices.** We will now use the following strategy to embed a permutation matrix into the output ring R of the homomorphic synthesizer S. Given a permutation matrix $N_{\ell,b} \in \{0,1\}^5$, its

ring-embedding $\mathbf{N}_{\ell,b}$ is defined as a $5m \times 5m$ matrix over the ring R of the form:

$$\mathbf{N}_{\ell,b} = \begin{bmatrix} \mathbf{N}_{\ell,b,1,1} & \dots & \mathbf{N}_{\ell,b,1,5} \\ \vdots & \ddots & \vdots \\ \mathbf{N}_{\ell,b,5,1} & \dots & \mathbf{N}_{\ell,b,5,5} \end{bmatrix},$$

where for each $w, v \in [5]$, $\mathbf{N}_{\ell,b,w,v} \in R^{m\times m}$ is as defined below:

$$\mathbf{N}_{\ell,b,w,v} = \begin{cases} \mathbf{0} & \text{if } N_{\ell,b}[w,v] = 0, \\ \text{uniformly random} & \text{otherwise.} \end{cases}$$

4. **Generating Guard Matrices.** We generate a sequence of "guard" matrix-pairs $\{(\mathbf{L}_\ell, \mathbf{R}_\ell)\}_{\ell\in[L]}$ (over the ring R underlying the homomorphic synthesizer) such that the following conditions hold:

   - For each $\ell \in [L]$, the "left guard" $\mathbf{L}_\ell$ is of dimension $c_1cm \times c_2m$ and the "right guard" $\mathbf{R}_\ell$ is of dimension $c_2m \times c_1m$ where c_1, c_2 are parameters such that $c_1 >> c_2$. Intuitively, each left guard will be very "tall and skinny" matrix, while each right guard is a "short and fat" matrix.
   - For each $\ell \in [L-1]$, we have $\mathbf{R}_\ell\mathbf{L}_{\ell+1} = \mathbf{D}_\ell \in R^{c_2m\times c_2m}$, where $\mathbf{D}_\ell$ is a "block-diagonal" matrix of the form
   $$\mathbf{D}_\ell = \mathsf{diag}(\mathbf{D}_{\ell,1}, \mathbf{D}_{\ell,2}, \dots, \mathbf{D}_{\ell,c_2m}),$$
   where for each $j \in [c_2m]$, $\mathbf{D}_{\ell,j}$ is a square matrix of dimension $m \times m$ over the ring R. More formally, suppose that for some $\ell \in [L-1]$, the guard matrices $\mathbf{R}_\ell$ and $\mathbf{L}_{\ell+1}$ have the following structure:

$$\mathbf{R}_\ell = \begin{bmatrix} -- & -- & \mathbf{R}_{\ell,1} & -- & -- \\ & & \vdots & & \\ -- & -- & \mathbf{R}_{\ell,c_2m} & -- & -- \end{bmatrix}, \quad \mathbf{L}_{\ell+1} = \begin{bmatrix} | & & | \\ | & & | \\ \mathbf{L}_{\ell+1,1} & \cdots & \mathbf{L}_{\ell+1,c_2m} \\ | & & | \\ | & & | \end{bmatrix}.$$

Then for each $j, j' \in [c_2m]$, we have

$$\mathbf{R}_{\ell,j}\mathbf{L}_{\ell,j'} = \begin{cases} \mathbf{D}_{\ell,j} & \text{if } j = j', \\ \mathbf{0} & \text{if } j \neq j'. \end{cases}$$

We now show that such a sequence of matrix-pairs can be created efficiently. For each $\ell \in [L-1]$ and parameter m as described above, do the following:

(a) Sample uniform matrices $\mathbf{A}_\ell \leftarrow R^{c_2m\times(c_1/2-c_2)m}$ and $\mathbf{B}_\ell \leftarrow R^{c_2m\times(c_1/2-c_2)m}$, and set

$$\mathbf{Y}_\ell = \begin{bmatrix} \mathbf{A}_\ell & \mathbf{J} \end{bmatrix}, \quad \mathbf{Z}_\ell = \begin{bmatrix} \mathbf{B}_\ell \\ \\ \mathbf{J}^{-1}(\mathbf{D} - \mathbf{A}_\ell\mathbf{B}_\ell) \end{bmatrix},$$

where $\mathbf{D}_\ell$ is a uniformly sampled "block-diagonal" matrix $R^{c_2m\times c_2m}$ as described above, and $\mathbf{J}$ is an *upper triangular* matrix in $R^{c_2m\times c_2m}$ with an efficiently computable inverse (we refer the reader to Section 5 for the details of how such a matrix can be efficiently sampled).

(b) Sample a uniform square matrix $\mathbf{X}_\ell \leftarrow R^{(c_1/2)m\times(c_1/2)m}$ and a uniform matrix $\mathbf{U}_\ell \leftarrow R^{c_2m\times(c_1/2)m}$, and set $\mathbf{R}_\ell$ and $\mathbf{L}_{\ell+1}$ as:

$$\mathbf{R}_\ell = \begin{bmatrix} \mathbf{U}_\ell & \mathbf{U}_\ell\mathbf{X}_\ell + \mathbf{Y}_\ell \end{bmatrix}, \quad \mathbf{L}_{\ell+1} = \begin{bmatrix} -\mathbf{X}_\ell\mathbf{Z}_\ell \\ \\ \mathbf{Z}_\ell \end{bmatrix}$$

It easy to see that for each $\ell \in [L-1]$, we have

$$\mathbf{R}_\ell \mathbf{L}_{\ell+1} = -\mathbf{U}_\ell \mathbf{X}_\ell \mathbf{Z}_\ell + (\mathbf{U}_\ell \mathbf{X}_\ell + \mathbf{Y}_\ell)\, \mathbf{Z}_\ell = \mathbf{Y}_\ell \mathbf{Z}_\ell = \mathbf{D}.$$

Note that we do not explicitly generate the "end-guard" matrices $\mathbf{L}_1$ and $\mathbf{R}_L$. For the moment, we assume that these guard matrices are set to the identity matrix $\mathbf{I}$ over $R^{m \times m}$.

5. **The Construction.** We are now ready to describe a simplified version of our iO construction. Given an oblivious branching program of depth L, the obfuscation algorithm does the following:

   (a) **Step 1:** Construct a sequence of ring-embedded permutation matrices of the form $\{\mathbf{N}_{\ell,b}\}_{\ell\in[L],b\in\{0,1\}}$ as described above.

   (b) **Step 2:** Generate a sequence of "guard" matrix-pairs $\{(\mathbf{L}_\ell, \mathbf{R}_\ell)\}_{\ell\in[L]}$ satisfying the constraints as described above, for parameters $c_2 = 5$ and $c_1 >> 5$.

   (c) **Step 3:** Generate a sequence of $2L$ "guarded" program matrix encodings $\{\widetilde{\mathbf{N}}_{\ell,b}\}_{\ell\in[L],b\in\{0,1\}}$, where $\widetilde{\mathbf{N}}_{\ell,b} = \mathbf{L}_\ell \mathbf{N}_{\ell,b} \mathbf{R}_\ell$.

6. **Evaluation and Zero Testing.** To efficiently evaluate the program on an input $\mathbf{x} = (x_1, \ldots, x_N)$, one can compute the following "subset-product" of the "guarded" program matrix encodings: $\mathbf{Q} = \prod_{\ell=1}^{L} \widetilde{\mathbf{N}}_{\ell,x_{\phi(\ell)}}$. It is easy to see that the final product $\mathbf{Q}$ is matrix of dimension $5m \times 5m$. Suppose, $\mathbf{Q} = \{\mathbf{Q}_{w,v}\}_{w,v\in[5]}$ where each $\mathbf{Q}_{w,v}$ has dimension $m \times m$.

   Also, let Q be the permutation matrix resulting from performing the same "subset-product" on the actual permutation matrices in the clear, i.e., let $Q = \prod_{\ell=1}^{L} \mathbf{M}_{\ell,x_{\phi(\ell)}}$. The zero test procedure crucially uses the following relation that holds with overwhelmingly large probability between each submatrix $\mathbf{Q}_{w,v}$ and the permutation matrix Q for any $(w,v) \in [5] \times [5]$:

   $$\mathbf{Q}_{w,v} = \mathbf{0} \quad \text{if and only if } Q[w,v] = 0.$$

   The zero test will then choose a single non-diagonal entry (i,j). Note that this entry is non-zero whenever the branching program outputs 0, i.e., whenever the matrix product Q is not the identity permutation matrix. Hence, it suffices to check whether the corresponding submatrix $\mathbf{Q}_{i,j}$ is zero.

Enforcing Consistency. We now augment the aforementioned construction to enforce input consistency. In other words, for a variable x_i that is associated with multiple levels of the program, the check should enforce that the same value of x_i is used at all of the associated levels. We handle this in our construction by making two main alterations to the previous construction:

1. **Generating Program-Carrying Matrices.** Recall that in the previous construction, at each level $\ell \in [L]$ and for each bit $b \in \{0,1\}$, we had a ring-embedded permutation matrix $\mathbf{N}_{\ell,b}$ of dimension $5m \times 5m$, structured as follows:

$$\mathbf{N}_{\ell,b} = \begin{bmatrix} \mathbf{N}_{\ell,b,1,1} & \cdots & \mathbf{N}_{\ell,b,1,5} \\ \vdots & \ddots & \vdots \\ \mathbf{N}_{\ell,b,5,1} & \cdots & \mathbf{N}_{\ell,b,5,5} \end{bmatrix},$$

For each level $\ell \in [L]$ and for each bit $b \in \{0,1\}$, we now construct a set of $5 \cdot 5$ block diagonal "program-carrying matrices" $\{\mathbf{P}_{\ell,b,w,v}\}_{w,v\in[5]}$, each of dimension $(2(L+N)+1)m \times (2(L+N)+1)m$, where each $\mathbf{P}_{\ell,b,w,v}$ is structured as:

$$\begin{bmatrix} \mathbf{N}_{\ell,b,w,v} & & & & & \\ & \mathbf{C}_{(\ell,1),(b,0),(w,v)} & & & & \\ & & \mathbf{C}_{(\ell,1),(b,1),(w,v)} & & & \\ & & & \ddots & & \\ & & & & \mathbf{C}_{(\ell,L+N),(b,0),(w,v)} & \\ & & & & & \mathbf{C}_{(\ell,L+N),(b,1),(w,v)} \end{bmatrix},$$

where for each $\ell \in [L]$, each $\ell' \in [L+N]$, each $b, b' \in \{0,1\}$ and each matrix position $(w,v) \in [5] \times [5]$, we have

$$\mathbf{C}_{(\ell,\ell'),(b,b'),(w,v)} = \begin{cases} \mathbf{0} & \text{if } (\ell', b') = (\ell, 1-b), \\ \text{uniform in } R^{m\times m} & \text{otherwise.} \end{cases}$$

2. **Generating Enforcer Matrices.** Next, for each variable index $i \in [N]$ and for each bit $b \in \{0,1\}$, we construct an additional set of block diagonal "enforcer matrices" $\mathbf{E}_{i,b}$ of dimension $(2(L+N)+1)m \times (2(L+N)+1)m$, structured as:

$$\mathbf{E}_{i,b} = \begin{bmatrix} \mathbf{T}_{i,b} & & & & & \\ & \mathbf{C}_{(i,1),(b,0)} & & & & \\ & & \mathbf{C}_{(i,1),(b,1)} & & & \\ & & & \ddots & & \\ & & & & \mathbf{C}_{(i,L+N),(b,0)} & \\ & & & & & \mathbf{C}_{(i,L+N),(b,1)} \end{bmatrix},$$

where for each $i \in [N]$, each $\ell' \in [L+N]$, and each $b, b' \in \{0,1\}$, we have $\mathbf{T}_{i,b} \leftarrow R^{m\times m}$, and

$$\mathbf{C}_{(\ell,\ell'),(b,b')} = \begin{cases} \mathbf{0} & \text{if } (i, b') = (\phi(\ell), b), \\ \text{uniform in } R^{m\times m} & \text{otherwise,} \end{cases}$$

where recall that $\phi : [L] \to [N]$ is a mapping such that for each $\ell \in [L]$, $i = \phi(\ell)$ indicates which variable x_i controls the ℓ^{th} level branch.

3. **Guarding Program-Carrying And Enforcer Matrices.** As before, we generate a sequence of "guard" matrix-pairs. We now need $2(L+N)$ guard matrices in order to cover both the program-carrying and enforcer matrices. More specifically, we generate a sequence of guard matrices $\{(\mathbf{L}_\ell, \mathbf{R}_\ell)\}_{\ell\in[L+N]}$ satisfying the same constraints, albeit for parameters $c_2 = 2(L+N)+1$ and $c_1 >> 2(L+N)+1$.

   Note that one difference from the simple iO construction presented earlier is in how we generate the "end-guard" matrices $\mathbf{L}_1$ and $\mathbf{R}_L$. In the simple construction, we assumed that these guard matrices were set to the identity matrix $\mathbf{I}$ over $R^{m\times m}$. For this construction, we structure them as multiple copies of the identity matrix "stacked" horizontally (for $\mathbf{L}_1$) and vertically (for $\mathbf{R}_L$), respectively.

   Finally, we output a sequence of $(5 \cdot 5 \cdot 2L + 2N)$ "guarded" matrix encodings:

$$\left\{ \widetilde{\mathbf{P}}_{\ell,b,w,v} = \mathbf{L}_\ell \mathbf{P}_{\ell,b,w,v} \mathbf{R}_\ell \right\}_{\ell\in[L+N], b\in\{0,1\}, w,v\in[5]}, \left\{ \widetilde{\mathbf{E}}_{i,b} = \mathbf{L}_{L+i} \mathbf{E}_{i,b} \mathbf{R}_{L+i} \right\}_{i\in[N], b\in\{0,1\}}.$$

Evaluation and Zero-testing. To efficiently evaluate the program on an input $\mathbf{x} = (x_1, \ldots, x_N)$, one can compute the following "subset-product" of the "guarded" program matrix encodings: $\mathbf{Q} = \prod_{\ell=1}^{L} \widetilde{\mathbf{P}}_{\ell,x_{\phi(\ell)}}$, where each $\widetilde{\mathbf{P}}_{\ell,x_{\phi(\ell)}}$ may be viewed as a 5×5 super-matrix of the corresponding program-carrying matrices $\{\widetilde{\mathbf{P}}_{\ell,x_{\phi(\ell)},w,v}\}_{w,v\in[5]}$. Suppose, $\mathbf{Q} = \{\mathbf{Q}_{w,v}\}_{w,v\in[5]}$.

The zero test chooses a single non-diagonal entry $\mathbf{Q}(w, v)$ and computes $\mathbf{Q}' = \mathbf{Q}_{w,v} \prod_{i=1}^{N} \widetilde{\mathbf{E}}_{L+i,x_i}$. Observe that $\mathbf{Q}'$ is a matrix of dimension $m \times m$. At this point, the zero test simply checks if $\mathbf{Q}' = \mathbf{0}$.

Correctness. We first observe that the consistency-check sub-matrices do not contribute to the result. To see this, observe the following:

1. For each (ℓ, b), if $b \neq x_\phi(\ell)$, then the consistency submatrices in the corresponding diagonal slot do not contribute to the final product since there is a zero submatrix in this diagonal slot in the program carrying matrix $\mathbf{P}_{\ell,x_{\phi(\ell)},w,v}$ for every $(w, v) \in [5] \times [5]$.

2. For each (ℓ, b), if $b = x_\phi(\ell)$, then the consistency submatrices in the corresponding diagonal slot do not contribute to the final product since there is a zero submatrix in this diagonal slot in the enforcer matrix $\mathbf{E}_{L+\phi(\ell),x_{\phi(\ell)}}$.

Thus the only potential contributions to the value of $\mathbf{Q}'$ come from the diagonal slots corresponding to the ring-embedded permutation matrices. It is now easy to see that $\mathbf{Q}' \neq \mathbf{0}$ if and only if the plaintext matrix subset-product Q has a non-zero entry in the position (i, j), indicating that the program P outputs 0 on input $\mathbf{x}$.

Parallel Program Execution. The next challenge towards our eventual goal of building iaiO is to be able to evaluate multiple oblivious branching programs in parallel on the same set of inputs (where each program has the same size and the same level-to-input mapping, but potentially differs in the contents of their matrices). We refer the reader to Section 5 for details on how to handle input activations for multiple programs being executed in parallel, and to Section 7 for the detailed proof of security.

2.6 A New iO Construction from "The MMap Strikes Back"

In [MZ18], Ma and Zhandry proposed a candidate asymmetric multilinear map scheme (referred to as the MZ18 MMap henceforth), that is based on the [CLT13] multilinear map, albeit with significant technical alterations to provably subvert zeroizing attacks [CLLT17]. To our knowledge, the MZ18 MMap is the only published candidate multilinear map that has not been publicly broken. In addition, the SXDH assumption plausibly holds for the MZ18 MMap.

We provide here a high-level overview of how the MZ18 MMap works . It encodes plaintexts from a ring R into matrices, where each such matrix encoding is associated with an interval level $[i, j]$ such that $i \leq j \leq N$ (N being the degree of the MMap). At a high level, two kinds of operations are defined over the MZ18 MMap encodings - addition and multiplication. The addition operation is only defined between encodings that belong to the same interval-level $[i, j]$, while the multiplication operation is only defined between encodings at "adjacent" interval-levels, i.e., it is only possible to multiply encodings at levels $[i, j]$ and $[j + 1, k]$ such that $i, j, k \in [N]$ and $i \leq j < k$, and this results in an encoding at the level $[i, \ell]$. Adding and multiplying encodings of plaintext elements a and b produces encodings of $(a + b)$ and $(a \cdot b)$, respectively, albeit at potentially different interval-levels.

In Section 12, we prove that the MZ18 MMap implies a secure slotted partial RKHwPRF, under the assumption that SXDH is hard over the MZ18 MMap encodings. This gives the first secure iO construction based on the MZ18 MMap. Below, we provide some intuition into how the instantiation is achieved.

First of all, our definitions of slotted partial RKHwPRF families only require that weak PRF outputs from different "slots" can only be multiplied in a fixed pre-determined order, which matches closely the "interval"-based restrictions imposed on multiplications of encodings by the authors of [MZ18], and allows us to use these intervals as slots. In addition, our definitions only require "partial" homomorphism, which accounts for the fact that the MZ18 MMap encodings are randomized and use "noisy" encodings from the [CLT13] MMap. We also show how to adopt the MZ18 zero test into an appropriate zero test for the slotted partial RKHwPRF family it implies.

Most importantly, though, our definitions and constructions are generic enough to encompass rings that are noncommutative. The MZ18 MMap uses matrix-based encodings of plaintext elements and crucially relies on the noncommutative nature of matrix multiplication to subvert zeriozing attacks; yet this has also made it difficult to instantiate iO from the MZ18 MMap using known techniques that are designed to work over fields or small-dimensional matrix rings, as opposed to generic matrix rings. Our key contribution is in developing techniques to work over matrix rings with arbitrarily large dimensions.

2.7 Negative Results and Discussion

Impossibility of FHS. We prove that field-embedded homomorphic synthesizers (FHS) are impossible to realize. This follows from two contradicting observations: first, that the subset sum indistinguishability we have been describing so far should hold due to the definition of the FHS, and second, that it cannot, due to the fact that we can efficiently compute inverses on the output field.

To see this, let $\mathbf{R} \in \mathcal{Y}^{2n \times n}$ be a matrix sampled randomly from the output space of an FHS. We let $\mathbf{R}_1, \mathbf{R}_2 \in \mathcal{Y}^{n \times n}$ be defined such that

$$\mathbf{R} = \left[\frac{\mathbf{R}_1}{\mathbf{R}_2} \right]$$

By what we have shown earlier and by the definition of an FHS, it is required that the tuple $(\mathbf{R}, \mathbf{RT})$ is indistinguishable from random for a randomly sampled $\mathbf{T} \leftarrow \mathcal{Y}^{n \times n}$. But since $\mathcal{Y}$ is a field, we can compute inverses (and pseudoinverses, if necessary) over both elements of $\mathcal{Y}$ and matrices of elements of $\mathcal{Y}$.

So given $(\mathbf{R}_1, \mathbf{R}_1\mathbf{T})$ and $(\mathbf{R}_2, \mathbf{R}_2\mathbf{T})$, we can compute $\mathbf{R}_1^{-1}$ and $\mathbf{R}_2^{-1}$ and check that

$$\mathbf{R}_1^{-1} (\mathbf{R}_1\mathbf{T}) = \mathbf{R}_2^{-1} (\mathbf{R}_2\mathbf{T}) .$$

On the other hand, this holds with negligible probability if $\mathbf{R}_1\mathbf{T}$ and $\mathbf{R}_2\mathbf{T}$ are replaced with truly random matrices. Thus we build a distinguisher that breaks the security of any FHS. This rules out the existence of both field-homomorphic synthesizers and field-homomorphic weak PRFs.

As we mentioned in the introduction, it does not seem likely that our attacks here can be extended to the ring case because it is not known how to compute kernels or inverses over general rings [ADM06, Jag12, YYHK18].

Cryptography and Mathematical Structure. We highlight that our results make progress towards completing the pathway from generic algebraically structured primitives to cryptographic applications that was originally initiated in [AMPR19, AMP19]. Specifically, by realizing iO from an RKHwPRF it seems that most of the well-known cryptographic primitives can be constructed in a generic manner from Minicrypt primitives endowed with additional algebraic structure. Most of the primitives that these existing frameworks relating mathematical structure to cryptography cannot handle seem to follow from assumptions that are even more structured than what we consider here.

Relationship to Generic/Idealized Models. A natural question to ask is what RKHwPRFs (or RHSs) offer in comparison with generic multilinear map or graded encoding models, which are also used to construct iO and NIKE (an analogous comparison would be KHwPRFs versus the generic group model [Sho97]). The generic/idealized models are useful for the purposes of abstraction and even more importantly, to prove lower bounds/negative results [MMN16, Ps16]. However, while a generic multilinear map or graded encoding is inherently limited from an instantiation point of view, it may be possible to securely instantiate an RKHwPRF. In other words, an RKHwPRF is a "standard-model" primitive, unlike generic graded encodings/multilinear maps.

In the same vein, iO constructions in the generic/idealized models are inherently unable to obfuscate primitive-dependent circuits since the "code" implementing a generic graded encoding/multilinear map is unavailable. By contrast, an iO construction based on an RKHwPRF is able to obfuscate any program that uses the "code" implementing the RKHwPRF itself. Moreover, cryptographic implications in the generic/idealized models can be too powerful; as a concrete example, virtual black-box obfuscation is realizable in the generic graded encoding model [BR14b, BGK$^+$14], but not in the standard model [BGI$^+$01].

Finally, RKHwPRFs are less structured than existing generic models: our constructions from RKHwPRFs work over any rings. To our knowledge, all traditional generic graded encoding models [BR14b, BR14a] require at least one large cyclic group in the ring or field that they use for computation, along with its explicit representation. We hope that relaxing such requirements might lead to new candidate constructions for iO/NIKE.

3 Preliminaries

3.1 Notation

For any positive integer n, we use $[n]$ to denote the set $\{1, \ldots, n\}$. For two positive integers m and n we denote the set $\{m, m+1, \ldots, n\}$ by $[m, n]$. We use λ for the security parameter. We use the symbols $\oplus$ and $\otimes$ as ring operations defined in the context. We assume that rings have multiplicative identity element. For a finite set S, we use $s \leftarrow S$ to sample uniformly from the set S.

Let $(R, \oplus, \otimes)$ be an arbitrary finite ring. We denote the additive/multiplicative identity of R by $0_R/1_R$. We define the multiplication of two matrices of ring elements in the natural way: for two arbitrary matrices

$$\mathbf{A} = [a_{ij}]_{\{i\in[\ell],j\in[m]\}} \in R^{\ell\times m} \quad , \quad \mathbf{B} = [b_{ij}]_{\{i\in[m],j\in[n]\}} \in R^{m\times n},$$

their product $\mathbf{C} = [c_{ij}]_{\{i\in[\ell],j\in[n]\}} = \mathbf{AB}$ is defined as

$$c_{ij} = (a_{i,1} \otimes b_{1,j}) \oplus (a_{i,2} \otimes b_{2,j}) \oplus \cdots \oplus (a_{i,m} \otimes b_{m,j}).$$

3.2 (Symmetric) Cryptographic Primitives

Weak Pseudorandom Functions. If $G : X \to Y$ is a function, let $G^{\$}$ denote a randomized oracle that, when invoked, samples $x \leftarrow X$ uniformly and outputs $(x, G(x))$. A keyed function family is a function $F : K \times X \to Y$ such that K is the key space and X, Y are input and output spaces, respectively. We may use the notation $F_k(x)$ to denote $F(k, x)$. A weak pseudorandom function (wPRF) family is an efficiently computable (keyed) function family F such that for all PPT adversaries $\mathcal{A}$ we have

$$\left|\Pr[\mathcal{A}^{F_k^{\$}} = 1] - \Pr[\mathcal{A}^{U^{\$}} = 1]\right| \leq \mathrm{negl}(\lambda),$$

where $k \leftarrow K$, and $U : X \to Y$ is a truly random function. Roughly speaking, the security requirement is that given access to polynomially many (random) input-output pairs of the form (x_i, y_i), no attacker can distinguish between the real experiment where $y_i = F_k(x_i)$ and the ideal experiment where $y_i = U(x_i)$ for a truly random function U.

(Pseudorandom) Synthesizers. Let ℓ and m be (polynomially bounded) integers, and let $S : X \times Y \to Z$ be an efficiently computable function. Assume that $\mathbf{x} \leftarrow X^\ell$ and $\mathbf{y} \in Y^m$ are two uniformly chosen vectors, and let $\mathbf{Z} \leftarrow Z^{\ell\times m}$ be a uniformly chosen matrix. We say that S is a pseudorandom synthesizer if for any probabilistic polynomial time (PPT) attacker we have

$$[S(\mathbf{x}, \mathbf{y})] \overset{c}{\approx} \mathbf{Z},$$

where $[S(\mathbf{x}, \mathbf{y})]$ is an $\ell \times m$ matrix whose ij^{th} entry is $S(x_i, y_j)$.

3.3 Homomorphic Primitives

We endow weak PRFs and (pseudorandom) synthesizers with ring homomorphism. We remark that it is also possible to define the notion of bounded homomorphism, similar to [AMPR19] and [AMP19] using a universal mapping that handles a bounded number of homomorphism. See [AMPR19] and [AMP19] for more details.

Definition 3.1. (Ring Key-Homomorphic Weak PRF.) A weak PRF family $F : K \times X \to Y$ is a Ring Key-Komomorphic weak PRF (RKHwPRF) family if it satisfies the following two properties:

- $(K, \oplus, \otimes)$ and $(Y, \boxplus, \boxtimes)$ are efficiently samplable (finite) rings with efficiently computable ring operations.
- For any $x \in X$ the function $F(\cdot, x) : K \to Y$ is a ring homomorphism, i.e., for any $x \in X$ and $k, k' \in K$ we have

$$F(k \oplus k', x) = F(k, x) \boxplus F(k', x) \quad , \quad F(k \otimes k', x) = F(k, x) \boxtimes F(k', x) \quad , \quad F(1_K, x) = 1_Y.$$

Definition 3.2. (Ring Input-Homomorphic Weak PRF.) A weak PRF family $F : K \times X \to Y$ is a Ring Input-Komomorphic weak PRF (RIHwPRF) family if it satisfies the following two properties:

- $(X, \oplus, \otimes)$ and $(Y, \boxplus, \boxtimes)$ are efficiently samplable (finite) rings with efficiently computable ring operations.
- For any $k \in K$ the function $F(k, \cdot) : X \to Y$ is a ring homomorphism, i.e., for any $k \in K$ and $x, x' \in X$ we have
$$F(k, x \oplus x') = F(k, x) \boxplus F(k, x') \quad , \quad F(k, x \otimes x') = F(k, x) \boxtimes F(k, x') \quad , \quad F(k, 1_X) = 1_Y .$$

Definition 3.3. (Ring-Embedded Homomorphic Synthesizer.) A Ring-Embedded Homomorphic Synthesizer $S : X \times G \to R$ is a synthesizer that satisfies the following properties:

- $(G, \oplus)$ is an efficiently samplable (finite) group with efficiently computable group operation.
- $(R, \boxplus, \boxtimes)$ is an efficiently samplable (finite) ring with efficiently computable ring operations.
- For any $x \in X$ the function $S(x, \cdot) : G \to R$ is a *group* homomorphism, i.e., for any $x \in X$ and $r_1, r_2 \in R$ we have
$$S(x, g_1 \oplus g_2) = S(x, r_1) \boxplus S(x, r_2)$$

It is easy to see that a ring-embedded homomorphic synthesizer is implied by an RKHwPRF or an RIHwPRF (for which the input space does not depend on the choice of the key).

3.4 Indistinguishability Obfuscation (iO)

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

Definition 3.4. 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 adversary $\mathcal{A}$ and for any two functionally equivalent circuits $C_1 \in \mathcal{C}_\lambda$ and $C_2 \in \mathcal{C}_\lambda$ such that $|C_0| = |C_1|$, it holds that:
$$|\Pr[\mathcal{A}(\lambda, C_0) = 1] - \Pr[\mathcal{A}(\lambda, C_1) = 1]| \leq \text{negl}(\lambda).$$

3.5 Input-Activated Obfuscation

In this part, we present a concise definition of an abstraction called input-activated obfuscation which is presented in [GLSW15]. We refer the reader to [GLSW15] for more details.

Let $P = \{P_1, \dots, P_\ell\}$ be a set of ℓ programs, where each program comes from a family $\mathcal{P}_\lambda$. Let M be an $n \times \ell \times 2$ dimensional matrix (array) of binary elements, where $M_{i,j,\beta} \in \{0,1\}$ for $i \in [n], j \in [\ell]$ and $\beta \in \{0,1\}$. We also let $i/j/\beta$ to denote the row, column, and "slot" of the matrix M. For each column of M we define a boolean function $f_j : \{0,1\}^n \to \{0,1\}$, where the program P_j is active on the input iff $f_j(x) = 1$. The function $f_j(x)$ is defined to 1 when $M_{i,j,x_i} = 1$ for *all* $i \in [n]$, and 0 otherwise.

An input-activated obfuscation is a pair of algorithms $\mathsf{Create}, \mathsf{Eval}$ where:

- Create: The creation algorithm takes λ, M, P as input and creates an input-activated obfuscation T.
- Eval: The evaluation algorithm takes an input-activated obfuscation T and an input $x \in \{0,1\}^n$, and outputs a bit.

Correctness. Let $S_x \subseteq [\ell]$ denote the set of all (column) indices j such that $f_j(x) = 1$. The correctness requires that if $S_x \neq \emptyset$ and $P_j = P_{j'}$ for all $j, j' \in S_x$, then $\mathsf{Eval}(T, x) = P_j(x)$ for all $j \in S_x$.

Inter-column Security. This game is parameterized by λ, M, P, two column indices j, k, a row index i^*, and a slot index β such that $M_{i^*,j,\beta} = 1$. We assume that program descriptions P_j and P_k are identical. We also assume that for every row $i \neq i^*$ and slot $\gamma = 1 - \beta$, if $M_{i,k,\gamma} = 1$ then $M_{i,j,\gamma} = 1$. (This means that the column j dominates column k)

The game proceeds as follows. First, the challenger samples a bit b. If $b = 0$, it runs $\mathsf{Create}(\lambda, M, P)$ to produce T. If $b = 1$, it forms M' by copying M and flipping the (i^*, k, β)th entry. It then gives T to the adversary where T is $\mathsf{Create}(\lambda, M', P)$. The advantage of the adversary is defined to be the probability that it guesses the bit b. An input-activated obfuscation satisfies inter-column security if the advantage of any PPT adversary in the mentioned game is negligible.

Intra-column Security. This game is parameterized by λ, M, P, an index j such that exists a row i^* where the corresponding slots are both 0, and two alternate columns C_0 and C_1 such that the i^* row has both slots equal to 0.

The game proceeds as follows. First, the challenger samples a bit b. If $b = 0$, it runs $\mathsf{Create}(\lambda, M, P)$ to produce T. If $b = 1$, it forms M' replacing jth column of M (in two slots) with C_0 and C_1 respectively. It then gives T to the adversary where T is $\mathsf{Create}(\lambda, M', P)$. The advantage of the adversary is defined to be the probability that it guesses the bit b. An input-activated obfuscation satisfies intra-column security if the advantage of any PPT adversary in the mentioned game is negligible.

Completely Inactive Program Security. This game is parameterized by λ, M, P, an alternate program P^*, and an index j such that jth column contains all zero entries in both slots.

The game proceeds as follows. First, the challenger samples a bit b. If $b = 0$, it runs $\mathsf{Create}(\lambda, M, P)$ to produce T. If $b = 1$, it forms P' by modifying P to replace the jth program with P^*. It then gives T to the adversary where T is $\mathsf{Create}(\lambda, M', P')$. The advantage of the adversary is defined to be the probability that it guesses the bit b. An input-activated obfuscation satisfies completely inactive program security if the advantage of any PPT adversary in the mentioned game is negligible.

Single-input Program Security. This game is parameterized by λ, M, P, an alternate program P^*, and an index j such that jth column of M corresponds to a point function f_j where f_j evaluates to 1 on a single input x^*, and $P^*(x^*) = P_j(x^*)$.

The game proceeds as follows. First, the challenger samples a bit b. If $b = 0$, it runs $\mathsf{Create}(\lambda, M, P)$ to produce T. If $b = 1$, it forms P' by modifying P to replace the jth program with P^*. It then gives T to the adversary where T is $\mathsf{Create}(\lambda, M', P')$. The advantage of the adversary is defined to be the probability that it guesses the bit b. An input-activated obfuscation satisfies single-input program switching security if the advantage of any PPT adversary in the mentioned game is negligible.

3.6 Leftover Hash Lemma

We consider the following lemmata which are related to the leftover hash lemma [IZ89], and its special cases over rings.

Lemma 3.5. *Let X_1 and X_2 be two independent and identically distributed random variables with finite support S. If $\Pr[X_1 = X_2] \leq (1 + 4\varepsilon^2)/|S|$, then the statistical distance between the uniform distribution over S and X_1 is at most ε.*

We remark that since the additive group of any ring is abelian, the following statement follows from uniformity (aka regularity) of subset sum over finite (abelian) groups.

Lemma 3.6. *Let R be a finite ring with additive/multiplicative identity $0_R/1_R$, and let $m > 3\log|R|$. Assume that $\mathbf{r} \leftarrow R^m$ is a vector of uniformly chosen ring elements. For any (unbounded) adversary we have*

$$(\mathbf{r}, \mathbf{r}^t\mathbf{s}) \overset{s}{\approx} (\mathbf{r}, \mathbf{u}),$$

where $\mathbf{u} \leftarrow R^m$ *is an* m*-dimensional vector of uniformly chosen ring elements and* $\mathbf{s} \leftarrow \{0_R, 1_R\}^m$.

We also need the following simple statement. A proof can be found in [Mic02].

Lemma 3.7. *Let* R *be a finite ring, and let* $\mathbf{r} = (r_1, \ldots, r_m)$ *be an arbitrary vector in* R^m*. If* $\mathbf{u} \leftarrow R^m$*, then the distribution of* $\mathbf{u}^t\mathbf{r}$ *(respectively,* $\mathbf{r}^t\mathbf{u}$*) is uniform over the left (respectively, right) ideal in* R *generated by the set* $(r_1, \ldots, r_m)$*.*

4 Noninteractive Multiparty Key Exchange

In this section, we show a construction of noninteractive multiparty key exchange from a ring-embedded homomorphic synthesizer. As we mentioned before, it is straightforward to show that a ring-homomorphic synthesizer is implied by either any RIHwPRF (for which the input space does not depend on the choice of the key) or any RKHwPRF. First, we mention a hardness assumption that is implied by ring-homomorphic synthesizers. The following theorem is adapting the Theorem 1 of [AMP19] to ring-embedded homomorphic synthesizers.

Theorem 4.1. *Let* $S : X \times G \to R$ *be a ring-embedded homomorphic synthesizer, and let* $m = \text{poly}(\lambda)$ *be an (arbitrary) positive integer. Let* $d = \text{poly}(\lambda)$ *be such that* $d > 3\log|G|$*. Let* $\mathbf{R} \leftarrow R^{m\times d}$ *be matrix of ring elements such that each entry* $r_{i,j}$ *(for* $i \in [m], j \in [d]$*) is drawn uniformly and independently from* R*. If* $\mathbf{s} \leftarrow \{0_R, 1_R\}^d$*, then for any PPT adversary we have*

$$(\mathbf{R}, \mathbf{R}\mathbf{s}) \overset{c}{\approx} (\mathbf{R}, \mathbf{u})$$

where $\mathbf{u} \leftarrow R^m$ *is a vector of uniformly chosen ring elements from* R*.*

Proof. The proof is almost identical to the proof of Theorem 1 of [AMP19], and we sketch an argument here. First, we define a matrix $\mathbf{M} \in R^{m\times d}$ as $\mathbf{M}_{i,j} = S(x_i, g_j)$, where $x_i \leftarrow X, g_j \leftarrow G$ (for $i \in [m], j \in [d]$) are chosen uniformly and independently. We also define the vector $\mathbf{g}$ as $\mathbf{g} = (g_1, \ldots, g_d)$. Now, we show that $(\mathbf{M}, \mathbf{M}\mathbf{s}) \overset{c}{\approx} (\mathbf{R}, \mathbf{u})$ where $\mathbf{R} \in R^{m\times d}$ (resp. $\mathbf{u} \in R^m$) is a uniformly chosen matrix (resp., vector) of ring elements. Using the homomorphism of S and by the leftover hash lemma over rings (Lemma 3.6) we can write

$$\mathbf{M}\mathbf{s} = \begin{pmatrix} S\left(x_1, \bigoplus_{\mathbf{s}} \mathbf{g}\right) \\ S\left(x_2, \bigoplus_{\mathbf{s}} \mathbf{g}\right) \\ \vdots \\ S\left(x_m, \bigoplus_{\mathbf{s}} \mathbf{g}\right) \end{pmatrix} \overset{s}{\approx} \begin{pmatrix} S\left(x_1, g^*\right) \\ S\left(x_2, g^*\right) \\ \vdots \\ S\left(x_m, g^*\right) \end{pmatrix},$$

where $g^* \leftarrow G$ is uniformly chosen. By the pseudorandomness property of S, we have $(\mathbf{M}, \mathbf{M}\mathbf{s}) \overset{c}{\approx} (\mathbf{R}, \mathbf{u})$. Observe that since $\mathbf{M} \overset{c}{\approx} \mathbf{R}$, a straightforward reduction implies that $(\mathbf{M}, \mathbf{M}\mathbf{s}) \overset{c}{\approx} (\mathbf{R}, \mathbf{R}\mathbf{s})$. By triangle inequality, it follows that $(\mathbf{R}, \mathbf{R}\mathbf{s}) \overset{c}{\approx} (\mathbf{R}, \mathbf{u})$, as required. □

Theorem 4.2. *Let* $S : X \times G \to R$ *be a ring-embedded homomorphic synthesizer, and let* $m = \text{poly}(\lambda)$ *be a positive integer such that* $m > 3\log|G|$*. Let* $M_m(R)$ *be the matrix ring over* R*, i.e., the ring of* m *by* m *square matrices over* R*. If* $F : M_m(R) \times M_m(R) \to M_m(R)$ *be the function defined by* $F(\mathbf{K}, \mathbf{X}) = \mathbf{X} \boxtimes \mathbf{K}$*, then* F *is a weak PRF (and hence a synthesizer). In addition,* F *satisfies (right)* $M_m(R)$*-module homomorphism over the key space, i.e., for any* $\mathbf{K}, \mathbf{K}', \mathbf{X} \in M_m(R)$ *we have*

$$F(\mathbf{K} \boxplus \mathbf{K}', \mathbf{X}) = F(\mathbf{K}, \mathbf{X}) \boxplus F(\mathbf{K}', \mathbf{X}) \quad , \quad F(\mathbf{K} \boxtimes \mathbf{K}', \mathbf{X}) = F(\mathbf{K}, \mathbf{X}) \boxtimes \mathbf{K}',$$

where $(\boxplus, \boxtimes)$ *is addition and multiplication over* $M_m(R)$*, respectively.*

Proof. Observe that (right) $\mathrm{M}_m(R)$-module homomorphism of F over the key space is easy to verify. We now prove the weak pseudorandomness of F. Let $Q = \text{poly}(\lambda)$ be any arbitrary positive integer. It is enough to show that

$$(\mathbf{A}, \mathbf{A}\mathbf{K}) \overset{c}{\approx} (\mathbf{A}, \mathbf{U}),$$

where $\mathbf{A} \leftarrow R^{Qm\times m}$ and $\mathbf{U} \leftarrow R^{Qm\times m}$. One can view $(\mathbf{A}, \mathbf{AK})$ as stacking up Q input-output pairs in the real (weak PRF) game. By Theorem 4.1, we have

$$(\mathbf{A}, \mathbf{As}) \overset{c}{\approx} (\mathbf{A}, \mathbf{u}),$$

where $\mathbf{s} \leftarrow \{0_R, 1_R\}^m$ and $\mathbf{u} \leftarrow R^m$. It is easy to see that if $\mathbf{k} \leftarrow R^m$, then the distributions of $\mathbf{k}$ and $\mathbf{s}+\mathbf{k}$ are identical, where $+$ denotes component-wise addition in R^m induced by R. It follows that

$$(\mathbf{A}, \mathbf{Ak}) \overset{s}{\approx} (\mathbf{A}, \mathbf{A}(\mathbf{k}+\mathbf{s})) \overset{c}{\approx} (\mathbf{A}, \mathbf{Ak}+\mathbf{u}) \overset{s}{\approx} (\mathbf{A}, \mathbf{u}'),$$

where $\mathbf{u}' \leftarrow R^m$. By applying a standard hybrid argument over the columns of $\mathbf{AK}$, and using the fact that $(\mathbf{A}, \mathbf{Ak}) \overset{c}{\approx} (\mathbf{A}, \mathbf{u})$, it follows that

$$(\mathbf{A}, \mathbf{AK}) \overset{c}{\approx} (\mathbf{A}, \mathbf{U}). \qquad \square$$

Noninteractive Three Party Key Exchange. Here we start with the simpler case of (noninteractive) three party key exchange protocol from any ring-embedded homomorphic synthesizer. Later, we show how to construct a noninteractive key exchange protocol for more than three parties, and we formally prove its security.

Given a ring-embedded homomorphic synthesizer $S : X \times G \to R$, we first fix parameters $m > 3\log|G|$ and $n > 6m^2 \log(|R|)$. Let $\mathcal{R} = M_m(R)$ denote m by m square matrices over R. We remark that $\log(|\mathcal{R}^{n\times n}|)$ is polynomial in the security parameter, and hence elements of $\mathcal{R}^{n\times n}$ can be represented using polynomially many bits.

We also assume that $\mathbf{R}^{(1)} \leftarrow \mathcal{R}^{n\times n}$ and $\mathbf{R}^{(2)} \leftarrow \mathcal{R}^{n\times n}$ are two matrices of uniformly chosen ring elements, and they are published as public parameters of the protocol. The protocol is described as follows:

- Alice generates her own (secret) randomness $\mathbf{S}_A \leftarrow \mathcal{R}^{n\times n}$, and publishes $\mathbf{P}_A := \mathbf{S}_A\mathbf{R}^{(1)}$.
- Bob chooses his randomness as $\mathbf{S}_B \leftarrow \mathcal{R}^{n\times n}$, and publishes $(\mathbf{P}_B^{(1)}, \mathbf{P}_B^{(2)})$ where
$$\mathbf{P}_B^{(1)} := \mathbf{R}^{(1)}\mathbf{S}_B \quad , \quad \mathbf{P}_B^{(2)} := \mathbf{S}_B\mathbf{R}^{(2)}.$$
- Charlies generates his randomness as $\mathcal{R}^{n\times n}$, and publishes $\mathbf{P}_C := \mathbf{R}^{(2)}\mathbf{S}_C$.
- The final shared secret is $\boxed{\mathbf{S}} := \mathbf{S}_A\mathbf{R}^{(1)}\mathbf{S}_B\mathbf{R}^{(2)}\mathbf{S}_C$. Alice/Bob/Charlie can compute the final secret $\boxed{\mathbf{S}}$ as
$$\begin{aligned}\boxed{\mathbf{S}} = \mathbf{S}_A\mathbf{R}^{(1)}\mathbf{S}_B\mathbf{R}^{(2)}\mathbf{S}_C &= \mathbf{S}_A\mathbf{P}_B^{(1)}\mathbf{P}_C && \text{(Alice)}\\ &= \mathbf{P}_A\mathbf{S}_B\mathbf{P}_C && \text{(Bob)}\\ &= \mathbf{P}_A\mathbf{P}_B^{(2)}\mathbf{S}_C && \text{(Charlie).}\end{aligned}$$

We formally prove the secruity of mentioned key exchange protocol via the following theorem:

Theorem 4.3. *Let $S : X \times G \to R$ be a ring-embedded homomorphic synthesizer, and assume that m and n be integers such that $m > 3\log|G|$ and $n > 6m^2\log(|R|)$. Let $\mathcal{R} = M_m(R)$ denote m by m square matrices matrices over R. If $\mathbf{R}^{(1)} \leftarrow \mathcal{R}^{n\times n}$ and $\mathbf{R}^{(2)} \leftarrow \mathcal{R}^{n\times n}$ are two matrices of uniformly chosen ring elements, for any PPT adversary we have*

$$\begin{aligned}&(\mathbf{R}^{(1)}, \mathbf{R}^{(2)}, \mathbf{S}_A\mathbf{R}^{(1)}, \mathbf{R}^{(1)}\mathbf{S}_B, \mathbf{S}_B\mathbf{R}^{(2)}, \mathbf{R}^{(2)}\mathbf{S}_C, \mathbf{S}_A\mathbf{R}^{(1)}\mathbf{S}_B\mathbf{R}^{(2)}\mathbf{S}_C)\\ \overset{c}{\approx}\ &(\mathbf{R}^{(1)}, \mathbf{R}^{(2)}, \mathbf{S}_A\mathbf{R}^{(1)}, \mathbf{R}^{(1)}\mathbf{S}_B, \mathbf{S}_B\mathbf{R}^{(2)}, \mathbf{R}^{(2)}\mathbf{S}_C, \mathbf{U})\end{aligned}$$

where $\mathbf{S}_A, \mathbf{S}_B, \mathbf{S}_C \leftarrow \mathcal{R}^{n\times n}$ are uniformly chosen (secret) matrices, and $\mathbf{U} \leftarrow \mathcal{R}^{n\times n}$.

Before explaining the proof, we show a few auxiliary lemmata that will be crucial for proving the security of the protocol.

Lemma 4.4. *Let R be a finite ring, and let $m > 6\log|R|$. For a vector $\mathbf{r} \in R^m$, let $\mathsf{LKer}(\mathbf{r})$ be the set of all vectors $\mathbf{w} \in R^m$ such that $\mathbf{w}^t\mathbf{r} = 0_R$. If $\mathbf{u} \leftarrow R^m$ and $\mathbf{r} \leftarrow R^m$, we have*

$$(\mathbf{r}, \mathbf{u}, \mathbf{v}^t\mathbf{u}) \overset{s}{\approx} (\mathbf{r}, \mathbf{u}, s),$$

where $\mathbf{v} \leftarrow \mathsf{LKer}(\mathbf{r})$ and $s \leftarrow R$.

Proof. We split the vectors as $\mathbf{u} = (\mathbf{u}_1, \mathbf{u}_2)$, $\mathbf{r} = (\mathbf{r}_1, \mathbf{r}_2)$, $\mathbf{v} = (\mathbf{v}_1, \mathbf{v}_2)$ such that $\mathbf{u}_2$, $\mathbf{r}_2$, and $\mathbf{v}_2$ all live in $R^{3\log(|R|)}$. By Lemma 3.6, it follows that if $\mathbf{r}_2$ is sampled uniformly, then (with overwhelming probability over the choice of $\mathbf{r}_2$) the (left) ideal generated by components of $\mathbf{r}_2$ is R, since otherwise the (left) ideal generated by $\mathbf{r}_2$ would not cover at least half of the elements in R (recall that any proper additive subgroup of R cannot contain more than half of the elements of R). Moreover, if $\mathbf{a}$ is sampled uniformly from $R^{3\log(|R|)}$ then $\mathbf{a}^t\mathbf{r}_2$ is (statistically close to) uniform over R . It follows that

$$(\mathbf{r}, \mathbf{v}_1, \mathbf{v}_2) \overset{s}{\approx} (\mathbf{r}, \mathbf{u}'_1, \mathbf{u}'_2),$$

where $\mathbf{u}'_1 \leftarrow R^{m-3\log(|R|)}$ is sampled uniformly and independently, and $\mathbf{u}'_2 \in R^{3\log(|R|)}$ is sampled conditioned on $\mathbf{u}_1^t\mathbf{r}_1 + \mathbf{u}_2^t\mathbf{r}_2 = 0_R$. This means that to generate a (statistically close to) uniform vector $\mathbf{v}$ in $\mathsf{LKer}(\mathbf{r})$, one can sample the first $m - 3\log(|R|)$ components (which is $\mathbf{v}_1$) uniformly, and generate the rest of the components (which is $\mathbf{v}_2$) conditioned on $\mathbf{v}_1^t\mathbf{r}_1 + \mathbf{v}_2^t\mathbf{r}_2 = 0_R$. In particular, this implies that first $m - 3\log(|R|) > 3\log(|R|)$ components of $\mathbf{v}$ generate R. By applying Lemma 3.7 and using the fact that components of $\mathbf{v}_1$ (and hence components of $\mathbf{v}$) generate R with overwhelming probability, it follows that

$$(\mathbf{r}, \mathbf{v}^t\mathbf{u}) \overset{s}{\approx} (\mathbf{r}, s).$$

Now we compute the collision probability for two *independent* instances of $(\mathbf{r}, \mathbf{u}, \mathbf{v}^t\mathbf{u})$ as

$$\begin{aligned}\Pr[(\mathbf{r}, \mathbf{u}, \mathbf{v}^t\mathbf{u}) = (\mathbf{r}', \mathbf{u}', \mathbf{v}'^t\mathbf{u}')] &= \Pr[\mathbf{v}^t\mathbf{u} = \mathbf{v}'^t\mathbf{u}' \mid \mathbf{r} = \mathbf{r}', \mathbf{u} = \mathbf{u}'] \cdot \Pr[\mathbf{r} = \mathbf{r}', \mathbf{u} = \mathbf{u}'] \\ &= \Pr[\mathbf{u}^t(\mathbf{v} - \mathbf{v}') = 0_R] \cdot |R|^{-2m} \\ &= \Pr[\mathbf{u}^t\mathbf{v} = 0_R] \cdot |R|^{-2m} \leq (1 + \text{negl}) \cdot |R|^{-2m-1},\end{aligned}$$

where the inequality follows from $(\mathbf{r}, \mathbf{v}^t\mathbf{u}) \overset{s}{\approx} (\mathbf{r}, s)$, and the last equality follows from the fact that distribution of $\mathbf{v} - \mathbf{v}'$ is identical to that of $\mathbf{v}$ (because $\mathsf{LKer}(\mathbf{r})$ forms an additive group). By applying Lemma 3.5, it follows that

$$(\mathbf{r}, \mathbf{u}, \mathbf{v}^t\mathbf{u}) \overset{s}{\approx} (\mathbf{r}, \mathbf{u}, s),$$

as required. □

We also need the following lemma. The proof is identical to the previous case.

Lemma 4.5. *Let R be a finite ring, and let $m > 6\log|R|$. For a vector $\mathbf{r} \in R^m$, let $\mathsf{RKer}(\mathbf{r})$ be the set of all vectors $\mathbf{w} \in R^m$ such that $\mathbf{r}^t\mathbf{w} = 0_R$. If $\mathbf{u} \leftarrow R^m$ and $\mathbf{r} \leftarrow R^m$, we have*

$$(\mathbf{r}, \mathbf{u}, \mathbf{u}^t\mathbf{v}) \overset{s}{\approx} (\mathbf{r}, \mathbf{u}, s),$$

where $\mathbf{v} \leftarrow \mathsf{RKer}(\mathbf{r})$ and $s \leftarrow R^m$.

Lemma 4.6. *Let R be a finite ring, and let $m > 6\log|R|$. If $\mathbf{r}, \mathbf{r}', \mathbf{u}, \mathbf{u}' \leftarrow R^m$ be four uniformly chosen vectors, and $\mathbf{S} \leftarrow R^{m\times m}$ be a uniformly chosen matrix of ring elements, we have*

$$(\mathbf{r}, \mathbf{r}', \mathbf{r}^t\mathbf{S}, \mathbf{S}\mathbf{r}', \mathbf{u}, \mathbf{u}', \mathbf{u}^t\mathbf{S}\mathbf{u}') \overset{s}{\approx} (\mathbf{r}, \mathbf{r}', \mathbf{r}^t\mathbf{S}, \mathbf{S}\mathbf{r}', \mathbf{u}, \mathbf{u}', s),$$

where $s \leftarrow R$ is a uniformly chosen single ring element.

Proof. Let $\mathbf{M} \in R^{m\times m}$ be a matrix such that each *column* of $\mathbf{M}$ is uniformly and independently chosen from $\mathsf{RKer}(\mathbf{r})$. Similarly, let $\mathbf{M}' \in R^{m\times m}$ be matrix such that each *row* of $\mathbf{M}'$ is uniformly and independently chosen from $\mathsf{LKer}(\mathbf{r}')$. Clearly, we have $\mathbf{r}^t\mathbf{M} = \mathbf{0}$ and $\mathbf{M}\mathbf{r}' = \mathbf{0}$. Observe that if $\mathbf{S}$ is a uniform matrix, then $\mathbf{S}$ and $\mathbf{S} + \mathbf{M}\mathbf{M}'$ are statistically indistinguishable, where $+$ denotes the matrix addition induced by R. By replacing $\mathbf{S}$ with $\mathbf{S} + \mathbf{M}\mathbf{M}'$, it is enough to show that

$$(\mathbf{r}, \mathbf{r}', \mathbf{r}^t\mathbf{S}, \mathbf{S}\mathbf{r}', \mathbf{u}, \mathbf{u}', \mathbf{u}^t\mathbf{S}\mathbf{u}' + \mathbf{u}^t\mathbf{M}\mathbf{M}'\mathbf{u}') \overset{s}{\approx} (\mathbf{r}, \mathbf{r}', \mathbf{r}^t\mathbf{S}, \mathbf{S}\mathbf{r}', \mathbf{u}, \mathbf{u}', s).$$

Now consider the term $\mathbf{u}^t\mathbf{M}\mathbf{M}'\mathbf{u}'$. Since both $\mathbf{M}$ and $\mathbf{M}'$ sampled independently from $\mathbf{S}$, it suffices to prove that

$$(\mathbf{r}, \mathbf{r}', \mathbf{u}, \mathbf{u}', \mathbf{u}^t\mathbf{M}\mathbf{M}'\mathbf{u}') \overset{s}{\approx} (\mathbf{r}, \mathbf{r}', \mathbf{u}, \mathbf{u}', s).$$

By Lemma 4.5, and a simple (statistical) hybrid argument over the columns of $\mathbf{M}$, it follows that

$$(\mathbf{r}, \mathbf{r}', \mathbf{u}, \mathbf{u}', \mathbf{u}^t\mathbf{M}\mathbf{M}'\mathbf{u}') \overset{s}{\approx} (\mathbf{r}, \mathbf{r}', \mathbf{u}, \mathbf{u}', \mathbf{v}^t\mathbf{M}'\mathbf{u}'),$$

where $\mathbf{v} \leftarrow R^m$ is a uniform vector. Similarly, by Lemma 4.4 and using a hybrid argument over the rows of $\mathbf{M}'$, we get

$$(\mathbf{r}, \mathbf{r}', \mathbf{u}, \mathbf{u}', \mathbf{v}^t\mathbf{M}'\mathbf{u}') \overset{s}{\approx} (\mathbf{r}, \mathbf{r}', \mathbf{u}, \mathbf{u}', \mathbf{v}^t\mathbf{v}'),$$

where $\mathbf{v}' \leftarrow R^m$. Since $\mathbf{v}$ and $\mathbf{v}'$ are uniform and independent of other terms, by Lemma 3.6 and Lemma 3.7 it follows that

$$(\mathbf{r}, \mathbf{r}', \mathbf{u}, \mathbf{u}', \mathbf{v}^t\mathbf{v}') \overset{s}{\approx} (\mathbf{r}, \mathbf{r}', \mathbf{u}, \mathbf{u}', s).$$

By triangle inequality, we conclude that

$$(\mathbf{r}, \mathbf{r}', \mathbf{r}^t\mathbf{S}, \mathbf{S}\mathbf{r}', \mathbf{u}, \mathbf{u}', \mathbf{u}^t\mathbf{S}\mathbf{u}') \overset{s}{\approx} (\mathbf{r}, \mathbf{r}', \mathbf{r}^t\mathbf{S}, \mathbf{S}\mathbf{r}', \mathbf{u}, \mathbf{u}', s).$$ □

Now we prove the following lemma, which may be viewed as a weaker version of Theorem 4.3 where we used vectors $\mathbf{s}_A$ and $\mathbf{s}_C$ (instead of matrices) as Alice's and Charlie's secrets, respectively.

Lemma 4.7. *Let $S : X \times G \to R$ be a ring-embedded homomorphic synthesizer, and assume that m and n be integers such that $m > 3\log|G|$ and $n > 6m^2\log(|R|)$. Let $\mathcal{R} = M_m(R)$ denote m by m square matrices over R. If $\mathbf{R}^{(1)} \leftarrow \mathcal{R}^{n\times n}$ and $\mathbf{R}^{(2)} \leftarrow \mathcal{R}^{n\times n}$ are two matrices of uniformly chosen ring elements, for any PPT adversary we have*

$$\begin{aligned}&(\mathbf{R}^{(1)}, \mathbf{R}^{(2)}, \mathbf{s}_A^t\mathbf{R}^{(1)}, \mathbf{R}^{(1)}\mathbf{S}_B, \mathbf{S}_B\mathbf{R}^{(2)}, \mathbf{R}^{(2)}\mathbf{s}_C, \mathbf{s}_A^t\mathbf{R}^{(1)}\mathbf{S}_B\mathbf{R}^{(2)}\mathbf{s}_C)\\ \overset{c}{\approx}\ &(\mathbf{R}^{(1)}, \mathbf{R}^{(2)}, \mathbf{s}_A^t\mathbf{R}^{(1)}, \mathbf{R}^{(1)}\mathbf{S}_B, \mathbf{S}_B\mathbf{R}^{(2)}, \mathbf{R}^{(2)}\mathbf{s}_C, u),\end{aligned}$$

where $\mathbf{s}_A \leftarrow \mathcal{R}^n, \mathbf{S}_B \leftarrow \mathcal{R}^{n\times n}, \mathbf{s}_C \leftarrow \mathcal{R}^n$, and $u \leftarrow \mathcal{R}$.

Proof. First, we define the following hybrids:

- $\mathcal{H}_0$: This corresponds to the "real" game, which is the tuple

$$(\mathbf{R}^{(1)}, \mathbf{R}^{(2)}, \mathbf{s}_A^t\mathbf{R}^{(1)}, \mathbf{R}^{(1)}\mathbf{S}_B, \mathbf{S}_B\mathbf{R}^{(2)}, \mathbf{R}^{(2)}\mathbf{s}_C, \mathbf{s}_A^t\mathbf{R}^{(1)}\mathbf{S}_B\mathbf{R}^{(2)}\mathbf{s}_C).$$

- $\mathcal{H}_1$ In this hybrid, we replace the vector $\mathbf{s}_A^t\mathbf{R}^{(1)}$ with a uniformly chosen vector $\mathbf{u}_1^t \leftarrow \mathcal{R}^n$, i.e., the corresponding tuple is

$$(\mathbf{R}^{(1)}, \mathbf{R}^{(2)}, \mathbf{u}_1^t, \mathbf{R}^{(1)}\mathbf{S}_B, \mathbf{S}_B\mathbf{R}^{(2)}, \mathbf{R}^{(2)}\mathbf{S}_C, \mathbf{u}_1^t\mathbf{S}_B\mathbf{R}^{(2)}\mathbf{s}_C).$$

- $\mathcal{H}_2$: In this hybrid, we replace $\mathbf{R}^{(2)}\mathbf{s}_C$ with a uniformly chosen vector $\mathbf{u}_2 \leftarrow \mathcal{R}^n$, i.e., the corresponding tuple is

$$(\mathbf{R}^{(1)}, \mathbf{R}^{(2)}, \mathbf{u}_1^t, \mathbf{R}^{(1)}\mathbf{S}_B, \mathbf{S}_B\mathbf{R}^{(2)}, \mathbf{u}_2, \mathbf{u}_1^t\mathbf{S}_B\mathbf{u}_2).$$

- $\mathcal{H}_3$: In this hybrid, we replace the term $\mathbf{u}_1^t\mathbf{S}_B\mathbf{u}_2$ with a uniform element $u \leftarrow \mathcal{R}$, i.e., the corresponding tuple is

$$(\mathbf{R}^{(1)}, \mathbf{R}^{(2)}, \mathbf{u}_1, \mathbf{R}^{(1)}\mathbf{S}_B, \mathbf{S}_B\mathbf{R}^{(2)}, \mathbf{u}_2, u).$$

- $\mathcal{H}_4$: In this hybrid, we replace $\mathbf{u}_1^t$ with $\mathbf{s}_A^t\mathbf{R}^{(1)}$, i.e., the corresponding tuple is
$$(\mathbf{R}^{(1)}, \mathbf{R}^{(2)}, \mathbf{s}_A^t\mathbf{R}^{(1)}, \mathbf{R}^{(1)}\mathbf{S}_B, \mathbf{S}_B\mathbf{R}^{(2)}, \mathbf{u}_2, u).$$
- $\mathcal{H}_5$: This corresponds to "ideal" game, and we replace $\mathbf{u}_2$ with $\mathbf{R}^{(2)}\mathbf{s}_C$. So the tuple is
$$(\mathbf{R}^{(1)}, \mathbf{R}^{(2)}, \mathbf{s}_A^t\mathbf{R}^{(1)}, \mathbf{R}^{(1)}\mathbf{S}_B, \mathbf{S}_B\mathbf{R}^{(2)}, \mathbf{R}^{(2)}\mathbf{s}_C, u).$$

Now we show that consecutive hybrids are indistinguishable, which implies the security of key exchange protocol.

- $\mathcal{H}_0 \overset{c}{\approx} \mathcal{H}_1$: By applying Theorem 4.1 and 4.2, if $\mathbf{R} \leftarrow \mathcal{R}^{n\times n}$ and $\mathbf{s} \leftarrow \mathcal{R}^n$ then we have $(\mathbf{R}, \mathbf{s}^t\mathbf{R}) \overset{c}{\approx} (\mathbf{R}, \mathbf{u}^t)$. Assuming there is an attacker $\mathcal{A}$ that distinguishes $\mathcal{H}_0$ and $\mathcal{H}_1$, we construct an attacker $\mathcal{B}$ that distinguishes $(\mathbf{R}, \mathbf{s}^t\mathbf{R})$ and $(\mathbf{R}, \mathbf{u}^t)$. Given a pair of the form $(\mathbf{R}, \mathbf{r}^t)$ (where $\mathbf{r}$ is either $\mathbf{s}^t\mathbf{R}$ or random), the reduction (uniformly) samples $\mathbf{R}^{(2)} \leftarrow \mathcal{R}^{n\times n}, \mathbf{S}_B \leftarrow \mathcal{R}^{n\times n}, \mathbf{s}_C \leftarrow \mathcal{R}^n$ and sets $\mathbf{R}^{(1)} := \mathbf{R}$. It then runs $\mathcal{A}$ on the following tuple
$$(\mathbf{R}^{(1)}, \mathbf{R}^{(2)}, \mathbf{r}^t, \mathbf{R}^{(1)}\mathbf{S}_B, \mathbf{S}_B\mathbf{R}^{(2)}, \mathbf{R}^{(2)}\mathbf{s}_C, \mathbf{r}^t\mathbf{S}_B\mathbf{R}^{(2)}\mathbf{s}_C).$$
Observe that if $\mathbf{r}^t = \mathbf{s}^t\mathbf{R}$, the tuple corresponds to $\mathcal{H}_0$. If $\mathbf{r}^t$ is random, the tuple corresponds to $\mathcal{H}_1$. Hence, the reduction perfectly simulates the consecutive hybrids. It follows that $\mathcal{H}_0 \overset{c}{\approx} \mathcal{H}_1$.
- $\mathcal{H}_1 \overset{c}{\approx} \mathcal{H}_2$: This is similar to the proof of $\mathcal{H}_0 \overset{c}{\approx} \mathcal{H}_1$.
- $\mathcal{H}_2 \overset{c}{\approx} \mathcal{H}_3$: For two vectors $\mathbf{x} \in \mathcal{R}^{n_1}$ and $\mathbf{y} \in \mathcal{R}^{n_2}$, let $F(\mathbf{x}, \mathbf{y})$ be an n_1 by n_2 matrix whose ij'th entry is x_iy_j. We remark that we use the same notation for row vectors as well, so clearly we have
$$F(\mathbf{x}, \mathbf{y}) = F(\mathbf{x}^t, \mathbf{y}^t) = F(\mathbf{x}^t, \mathbf{y}) = F(\mathbf{x}, \mathbf{y}^t).$$
By Theorem 4.2, we know that $F(\mathbf{x}, \mathbf{y})$ is computationally indistinguishable from a uniform matrix $\mathbf{U} \in \mathcal{R}^{n_1\times n_2}$. Let $\mathbf{x}, \mathbf{y}, \mathbf{r}_1, \mathbf{r}_2 \leftarrow \mathcal{R}^n$ be four uniformly chosen vectors. Since statistical distance cannot be increased by applying a (randomized) function, by Lemma 4.6 it follows that
$$\begin{aligned}&\big(F(\mathbf{x}, \mathbf{r}), F(\mathbf{r}', \mathbf{y}), \mathbf{u}_1, F(\mathbf{x}, \mathbf{r}^t\mathbf{S}_B), F(\mathbf{S}_B\mathbf{r}', \mathbf{y}), \mathbf{u}_2, \mathbf{u}_1^t\mathbf{S}_B\mathbf{u}_2\big)\\ \overset{s}{\approx}\ &\big(F(\mathbf{x}, \mathbf{r}), F(\mathbf{r}', \mathbf{y}), \mathbf{u}_1, F(\mathbf{x}, \mathbf{r}^t\mathbf{S}_B), F(\mathbf{S}_B\mathbf{r}', \mathbf{y}), \mathbf{u}_2, u\big).\end{aligned}$$
Using $\mathcal{R}$-module homomorphism of F we get
$$\begin{aligned}&\big(F(\mathbf{x}, \mathbf{r}), F(\mathbf{r}', \mathbf{y}), \mathbf{u}_1, F(\mathbf{x}, \mathbf{r})\mathbf{S}_B, \mathbf{S}_B F(\mathbf{y}, \mathbf{r}'), \mathbf{u}_2, \mathbf{u}_1^t\mathbf{S}_B\mathbf{u}_2\big)\\ \overset{s}{\approx}\ &\big(F(\mathbf{x}, \mathbf{r}), F(\mathbf{r}', \mathbf{y}), \mathbf{u}_1, F(\mathbf{x}, \mathbf{r})\mathbf{S}_B, \mathbf{S}_B F(\mathbf{y}, \mathbf{r}'), \mathbf{u}_2, u\big).\end{aligned}$$
By Theorem 4.2, we know that $(F(\mathbf{x}, \mathbf{r}), F(\mathbf{r}', \mathbf{y})) \overset{c}{\approx} (\mathbf{R}^{(1)}, \mathbf{R}^{(2)})$ where $\mathbf{R}^{(1)}, \mathbf{R}^{(2)} \leftarrow \mathcal{R}^{n\times n}$. By plugging in the corresponding terms, it follows that
$$(\mathbf{R}^{(1)}, \mathbf{R}^{(2)}, \mathbf{u}_1^t, \mathbf{R}^{(1)}\mathbf{S}_B, \mathbf{S}_B\mathbf{R}^{(2)}, \mathbf{u}_2, \mathbf{u}_1^t\mathbf{S}_B\mathbf{u}_2) \overset{c}{\approx} (\mathbf{R}^{(1)}, \mathbf{R}^{(2)}, \mathbf{u}_1^t, \mathbf{R}^{(1)}\mathbf{S}_B, \mathbf{S}_B\mathbf{R}^{(2)}, \mathbf{u}_2, u).$$
- $\mathcal{H}_3 \overset{c}{\approx} \mathcal{H}_4$: This is similar to the proof of $\mathcal{H}_0 \overset{c}{\approx} \mathcal{H}_1$.
- $\mathcal{H}_4 \overset{c}{\approx} \mathcal{H}_5$: This is similar to the proof of $\mathcal{H}_0 \overset{c}{\approx} \mathcal{H}_1$. □

Proof of Theorem 4.3. The idea is similar to the proof of $\mathcal{H}_2 \overset{c}{\approx} \mathcal{H}_3$ in the previous lemma. By Lemma 4.7, we know that
$$\begin{aligned}&\Big(\mathbf{R}^{(1)}, \mathbf{R}^{(2)}, \mathbf{s}_A^t\mathbf{R}^{(1)}, \mathbf{R}^{(1)}\mathbf{S}_B, \mathbf{S}_B\mathbf{R}^{(2)}, \mathbf{R}^{(2)}\mathbf{s}_C, \mathbf{s}_A^t\mathbf{R}^{(1)}\mathbf{S}_B\mathbf{R}^{(2)}\mathbf{s}_C\Big)\\ \overset{c}{\approx}\ &\Big(\mathbf{R}^{(1)}, \mathbf{R}^{(2)}, \mathbf{s}_A^t\mathbf{R}^{(1)}, \mathbf{R}^{(1)}\mathbf{S}_B, \mathbf{S}_B\mathbf{R}^{(2)}, \mathbf{R}^{(2)}\mathbf{s}_C, u\Big).\end{aligned}$$

Let $\mathbf{x} \leftarrow \mathcal{R}^m$ be a uniform vector. Since $F(\mathbf{s}_A^t \mathbf{R}^{(1)}, \mathbf{x})$ and $F(\mathbf{x}, u)$ can be computed in polynomial time, it follows that

$$\begin{aligned}&\Big(\mathbf{R}^{(1)}, \mathbf{R}^{(2)}, F(\mathbf{x}, \mathbf{s}_A^t \mathbf{R}^{(1)}), \mathbf{R}^{(1)}\mathbf{S}_B, \mathbf{S}_B\mathbf{R}^{(2)}, \mathbf{R}^{(2)}\mathbf{s}_C, F(\mathbf{x}, \mathbf{s}_A^t \mathbf{R}^{(1)}\mathbf{S}_B\mathbf{R}^{(2)}\mathbf{s}_C)\Big)\\ \overset{c}{\approx}\ &\Big(\mathbf{R}^{(1)}, \mathbf{R}^{(2)}, F(\mathbf{x}, \mathbf{s}_A^t \mathbf{R}^{(1)}), \mathbf{R}^{(1)}\mathbf{S}_B, \mathbf{S}_B\mathbf{R}^{(2)}, \mathbf{R}^{(2)}\mathbf{s}_C, F(\mathbf{x}, u)\Big).\end{aligned}$$

Using $\mathcal{R}$-module homomorphism of F we get

$$\begin{aligned}&\Big(\mathbf{R}^{(1)}, \mathbf{R}^{(2)}, F(\mathbf{x}, \mathbf{s}_A^t)\mathbf{R}^{(1)}, \mathbf{R}^{(1)}\mathbf{S}_B, \mathbf{S}_B\mathbf{R}^{(2)}, \mathbf{R}^{(2)}\mathbf{s}_C, F(\mathbf{x}, \mathbf{s}_A^t)\mathbf{R}^{(1)}\mathbf{S}_B\mathbf{R}^{(2)}\mathbf{s}_C\Big)\\ \overset{c}{\approx}\ &\Big(\mathbf{R}^{(1)}, \mathbf{R}^{(2)}, F(\mathbf{x}, \mathbf{s}_A^t)\mathbf{R}^{(1)}, \mathbf{R}^{(1)}\mathbf{S}_B, \mathbf{S}_B\mathbf{R}^{(2)}, \mathbf{R}^{(2)}\mathbf{s}_C, F(\mathbf{x}, u)\Big).\end{aligned}$$

By Theorem 4.2, we know that $(F(\mathbf{s}_A^t, \mathbf{x}), F(\mathbf{x}, u)) \overset{c}{\approx} (\mathbf{S}_A, \mathbf{u}^t)$ where $\mathbf{S}_A \leftarrow \mathcal{R}^{n\times n}$ and $\mathbf{u} \leftarrow \mathcal{R}^n$. By plugging in the corresponding terms, it follows that

$$\begin{aligned}&\Big(\mathbf{R}^{(1)}, \mathbf{R}^{(2)}, \mathbf{S}_A\mathbf{R}^{(1)}, \mathbf{R}^{(1)}\mathbf{S}_B, \mathbf{S}_B\mathbf{R}^{(2)}, \mathbf{R}^{(2)}\mathbf{s}_C, \mathbf{S}_A\mathbf{R}^{(1)}\mathbf{S}_B\mathbf{R}^{(2)}\mathbf{s}_C\Big)\\ \overset{c}{\approx}\ &\Big(\mathbf{R}^{(1)}, \mathbf{R}^{(2)}, \mathbf{S}_A\mathbf{R}^{(1)}, \mathbf{R}^{(1)}\mathbf{S}_B, \mathbf{S}_B\mathbf{R}^{(2)}, \mathbf{R}^{(2)}\mathbf{s}_C, \mathbf{u}\Big).\end{aligned}$$

By a similar argument if $\mathbf{y} \leftarrow \mathcal{R}^n$, we have

$$\begin{aligned}&\Big(\mathbf{R}^{(1)}, \mathbf{R}^{(2)}, \mathbf{S}_A\mathbf{R}^{(1)}, \mathbf{R}^{(1)}\mathbf{S}_B, \mathbf{S}_B\mathbf{R}^{(2)}, \mathbf{R}^{(2)}F(\mathbf{s}_C, \mathbf{y}), \mathbf{S}_A\mathbf{R}^{(1)}\mathbf{S}_B\mathbf{R}^{(2)}F(\mathbf{s}_C, \mathbf{y})\Big)\\ \overset{c}{\approx}\ &\Big(\mathbf{R}^{(1)}, \mathbf{R}^{(2)}, \mathbf{S}_A\mathbf{R}^{(1)}, \mathbf{R}^{(1)}\mathbf{S}_B, \mathbf{S}_B\mathbf{R}^{(2)}, \mathbf{R}^{(2)}F(\mathbf{s}_C, \mathbf{y}), F(\mathbf{u}, \mathbf{y})\Big).\end{aligned}$$

By Theorem 4.2, we know that $(F(\mathbf{s}_C, \mathbf{y}), F(\mathbf{u}, \mathbf{y})) \overset{c}{\approx} (\mathbf{S}_C, \mathbf{U})$ where $\mathbf{S}_A \leftarrow \mathcal{R}^{n\times n}$ and $\mathbf{U} \leftarrow \mathcal{R}^{n\times n}$. By plugging in the corresponding terms, it follows that

$$\begin{aligned}&(\mathbf{R}^{(1)}, \mathbf{R}^{(2)}, \mathbf{S}_A\mathbf{R}^{(1)}, \mathbf{R}^{(1)}\mathbf{S}_B, \mathbf{S}_B\mathbf{R}^{(2)}, \mathbf{R}^{(2)}\mathbf{S}_C, \mathbf{S}_A\mathbf{R}^{(1)}\mathbf{S}_B\mathbf{R}^{(2)}\mathbf{S}_C)\\ \overset{c}{\approx}\ &(\mathbf{R}^{(1)}, \mathbf{R}^{(2)}, \mathbf{S}_A\mathbf{R}^{(1)}, \mathbf{R}^{(1)}\mathbf{S}_B, \mathbf{S}_B\mathbf{R}^{(2)}, \mathbf{R}^{(2)}\mathbf{S}_C, \mathbf{U}),\end{aligned}$$

and the proof is complete. □

Generalizing to Any Number of Parties. Now we describe a (noninteractive) k-party key exchange protocol for any k. Similar to the three-party case, let $S : X \times G \to R$ be a ring-embedded homomorphic synthesizer, and assume that m and n be integers such that $m > 3\log|G|$ and $n > 6m^2\log(|R|)$. Let $\mathcal{R} = M_m(R)$ denote m by m square matrices matrices over R, and let $\mathbf{R}^{(1)}, \ldots, \mathbf{R}^{(k-1)}$ be $k-1$ matrices that are uniformly chosen from $\mathcal{R}^{n\times n}$ (published as public parameters). The protocol is described as follows:

- Party 1 chooses its randomness $\mathbf{S}_1 \leftarrow \mathcal{R}^{n\times n}$, and publishes $\mathbf{P}_1 = \mathbf{S}_1\mathbf{R}^{(1)}$.
- Each party i (for $2 \le i \le k-1$) chooses its randomness $\mathbf{S}_i \leftarrow \mathcal{R}^{n\times n}$, and publishes $(\mathbf{P}_i^{(1)}, \mathbf{P}_i^{(2)})$ where
$$\mathbf{P}_i^{(1)} = \mathbf{R}^{(i-1)}\mathbf{S}_i \quad , \quad \mathbf{P}_i^{(2)} = \mathbf{S}_i\mathbf{R}^{(i)}.$$
- Party k chooses its randomness $\mathbf{S}_k \leftarrow \mathcal{R}^{n\times n}$, and publishes $\mathbf{P}_k = \mathbf{R}^{(k-1)}\mathbf{S}_k$.
- The final shared secret is $\boxed{\mathbf{S}} = \mathbf{S}_1\mathbf{R}^{(1)}\mathbf{S}_2\mathbf{R}^{(2)}\cdots\mathbf{S}_{k-1}\mathbf{R}^{(k-1)}\mathbf{S}_k$. Parties can compute the final secret $\boxed{\mathbf{S}}$ as
$$\begin{aligned}\boxed{\mathbf{S}} &= \mathbf{S}_1\mathbf{P}_2^{(1)}\mathbf{P}_3^{(1)}\cdots\mathbf{P}_{k-1}^{(1)}\mathbf{P}_k && \text{(Party 1)}\\ &= \mathbf{P}_1\mathbf{P}_2^{(2)}\cdots\mathbf{P}_{i-1}^{2}\mathbf{S}_i\mathbf{P}_{i+1}^{(1)}\cdots\mathbf{P}_{k-1}^{(1)}\mathbf{P}_k && \text{(Party } i \text{ for } 2 \le i \le k-1)\\ &= \mathbf{P}_1\mathbf{P}_2^{(2)}\mathbf{P}_3^{(2)}\cdots\mathbf{P}_{k-1}^{(2)}\mathbf{S}_k && \text{(Party } k).\end{aligned}$$

The security proof for the aforementioned protocol is similar to the proof of 4.3, and we sketch an argument here. Let the following matrices

$$\left(\{\mathbf{S}_i\}_{i\in[k]}, \{\mathbf{R}^{(i)}\}_{i\in[k-1]}, \mathbf{P}_1, \{\mathbf{P}_i^{(1)}, \mathbf{P}_i^{(2)}\}_{i\in[k-1]}, \mathbf{P}_k, \boxed{\mathbf{S}}\right),$$

be defined as in the protocol. It is enough to show that

$$\begin{aligned}&\left(\{\mathbf{R}^{(i)}\}_{i\in[k-1]}, \mathbf{P}_1, \{\mathbf{P}_i^{(1)}, \mathbf{P}_i^{(2)}\}_{i\in[2,k-1]}, \mathbf{P}_k, \boxed{\mathbf{S}}\right)\\ \overset{c}{\approx}&\left(\{\mathbf{R}^{(i)}\}_{i\in[k-1]}, \mathbf{P}_1, \{\mathbf{P}_i^{(1)}, \mathbf{P}_i^{(2)}\}_{i\in[2,k-1]}, \mathbf{P}_k, \mathbf{U}\right)\end{aligned}$$

where $\mathbf{U} \leftarrow \mathcal{R}^{n\times n}$ is a uniform matrix. The security proof is similar to the three party case, and we sketch an argument here. First, observe that similar to the three-party case, it is sufficient to prove the following "inefficient" version of the protocol

$$\begin{aligned}&\left(\{\mathbf{R}^{(i)}\}_{i\in[k-1]}, \mathbf{P}_1, \{\mathbf{P}_i^{(1)}, \mathbf{P}_i^{(2)}\}_{i\in[2,k-1]}, \mathbf{R}^{(k-1)}\mathbf{s}_k, \mathbf{S}_1\mathbf{R}^{(1)}\cdots\mathbf{S}_{k-1}\mathbf{R}^{(k-1)}\mathbf{s}_k\right)\\ \overset{c}{\approx}&\left(\{\mathbf{R}^{(i)}\}_{i\in[k-1]}, \mathbf{P}_1, \{\mathbf{P}_i^{(1)}, \mathbf{P}_i^{(2)}\}_{i\in[2,k-1]}, \mathbf{R}^{(k-1)}\mathbf{s}_k, \mathbf{u}\right),\end{aligned}$$

where kth party used a vector (instead of a matrix) as its secret. To prove the latter, first we replace $\mathbf{R}^{(k-1)}\mathbf{s}_k$ with a uniform vector $\mathbf{u}'$. We then replace $\mathbf{R}^{(k-1)}$ with $F(\mathbf{r},\mathbf{x})$ where $\mathbf{r},\mathbf{x}$ are uniform vectors in $\mathcal{R}^n$. By Theorem 4.2, we need to prove that

$$\begin{aligned}&\left(\{\mathbf{R}^{(i)}\}_{i\in[k-2]}, F(\mathbf{r},\mathbf{x}), \mathbf{P}_1, \{\mathbf{P}_i^{(1)}, \mathbf{P}_i^{(2)}\}_{i\in[2,k-1]}, \mathbf{u}', \mathbf{S}_1\mathbf{R}^{(1)}\cdots\mathbf{S}_{k-1}\mathbf{u}'\right)\\ \overset{c}{\approx}&\left(\{\mathbf{R}^{(i)}\}_{i\in[k-2]}, F(\mathbf{r},\mathbf{x}), \mathbf{P}_1, \{\mathbf{P}_i^{(1)}, \mathbf{P}_i^{(2)}\}_{i\in[2,k-1]}, \mathbf{u}', \mathbf{u}\right),\end{aligned}$$

and hence it is enough to show that

$$\begin{aligned}&\left(\{\mathbf{R}^{(i)}\}_{i\in[1,k-2]}, \mathbf{r}, \mathbf{P}_1, \{\mathbf{P}_i^{(1)}, \mathbf{P}_i^{(2)}\}_{i\in[,k-2]}, \mathbf{R}^{(k-2)}\mathbf{S}_{k-1}, \mathbf{S}_{k-1}\mathbf{r}, \mathbf{u}', \mathbf{S}_1\mathbf{R}^{(1)}\cdots\mathbf{R}^{(k-2)}\mathbf{S}_{k-1}\mathbf{u}'\right)\\ \overset{c}{\approx}&\left(\{\mathbf{R}^{(i)}\}_{i\in[1,k-2]}, \mathbf{r}, \mathbf{P}_1, \{\mathbf{P}_i^{(1)}, \mathbf{P}_i^{(2)}\}_{i\in[,k-2]}, \mathbf{R}^{(k-2)}\mathbf{S}_{k-1}, \mathbf{S}_{k-1}\mathbf{r}, \mathbf{u}', \mathbf{u}\right).\end{aligned}$$

Observe that if $\mathbf{S}_{k-1}\mathbf{r}$ was not present in the tuples above, then the computational indistinguishability of two tuples would follow from security of $(k-1)$-party key exchange protocol. To get around this problem, we replace $\mathbf{R}^{(k-2)}$ with $F(\mathbf{r}',\mathbf{y})$ where $\mathbf{r}'$ and $\mathbf{y}$ are sampled uniformly and independently from $\mathcal{R}^n$. We also replace $\mathbf{S}_{k-1}$ with $\mathbf{S}_{k-1}+\mathbf{M}$ where $\mathbf{M} \in \mathcal{R}^{n\times n}$ is a matrix whose columns uniformly and independently sampled from $\mathsf{RKer}(\mathbf{y})$. By Lemma 4.4, the term $(\mathbf{S}_{k-1}+\mathbf{M})\mathbf{r}$ will be uniform and independent of other components of the tuple. On the other hand, $\mathbf{S}_{k-1}$ and $\mathbf{S}_{k-1}+\mathbf{M}$ are statistically indistinguishable. It follows that

$$\begin{aligned}&\left(\{\mathbf{R}^{(i)}\}_{i\in[1,k-2]}, \mathbf{r}, \mathbf{P}_1, \{\mathbf{P}_i^{(1)}, \mathbf{P}_i^{(2)}\}_{i\in[,k-2]}, \mathbf{R}^{(k-2)}\mathbf{S}_{k-1}, \hat{\mathbf{u}}, \mathbf{u}', \mathbf{S}_1\mathbf{R}^{(1)}\cdots\mathbf{R}^{(k-2)}\mathbf{S}_{k-1}\mathbf{u}'\right)\\ \overset{c}{\approx}&\left(\{\mathbf{R}^{(i)}\}_{i\in[1,k-2]}, \mathbf{r}, \mathbf{P}_1, \{\mathbf{P}_i^{(1)}, \mathbf{P}_i^{(2)}\}_{i\in[,k-2]}, \mathbf{R}^{(k-2)}\mathbf{S}_{k-1}, \hat{\mathbf{u}}, \mathbf{u}', \mathbf{u}\right),\end{aligned}$$

where $\hat{\mathbf{u}}$ is uniform and independent of any other randomness. It is easy to see that the tuples above are computationally indistinguishable based on the security of $(k-1)$-party key exchange protocol. The rest of the proof is almost identical to 3-party case, and hence we omit the details.

Remark 4.8. We remark that in the constructions and proofs above, we never used the fact that the output ring R of the ring-embedded homomorphic synthesizer is commutative. The reader may note that for any nontrivial ring R, the matrix ring $M_n(R)$ for any $n > 2$ is noncommutative. Therefore, all the constructions inherently rely on noncommutative *matrix rings*, and hence some of the known algorithms to solve a system of linear equations over certain commutative rings are not applicable here.

5 Input-Activated Indistinguishability Obfuscator

In this section, we show how to construct an indistinguishability obfuscator for $\mathcal{NC}^1$ from any ring-embedded homomorphic synthesizer. Given a ring-embedded homomorphic synthesizer $S : X \times G \to R$, we fix appropriately large parameter m. Let $\mathbf{I}$ and $\mathbf{0}$ denote the identity matrix and all-zero matrix of dimension $m \times m$ over the ring R, respectively.

5.1 Core iO Construction

Permutation Branching Programs. Throughout this section, we assume that any $\mathcal{NC}^1$ program is represented as an oblivious permutation branching program P of width 5. Let $P = \{M_{\ell,0}, M_{\ell,1}\}_{\ell \in [L]}$, where L denotes the depth of P and each $M_{\ell,b} \in \{0,1\}^{5\times 5}$ is a permutation matrix. Also, let $x_1, \ldots, x_N$ denote the input variables, and let $\phi : [L] \to [N]$ be a mapping such that for each $\ell \in [L]$, $i = \phi(\ell)$ indicates which variable x_i controls the ℓ^{th} level branch.

Killian-Style Randomization. We uniformly randomly sample *additional permutation matrices* $Z_1, \ldots, Z_{L-1} \in \{0,1\}^{5\times 5}$, and define a new set of matrices $\{N_{\ell,b}\}_{\ell\in[L],b\in\{0,1\}}$, where

$$N_{\ell,b} = Z_{\ell-1}^{-1} M_{\ell,b} Z_\ell,$$

where Z_0 and Z_L are both set to be the identity permutation. Note that the set of all permutation matrices over $\{0,1\}^{5\times 5}$ form a group, which has the following implications:

- Each permutation matrix Z_ℓ is efficiently invertible.
- Each resulting matrix $N_{\ell,b}$ is also a permutation matrix.
- The aforementioned randomization technique information-theoretically hides the original set of permutation matrices, while retaining the program behavior as is.

Ring-Embedding Permutation Matrices. Throughout this section, we will use the following strategy to embed a permutation matrix into the output ring R of the homomorphic synthesizer S. Given a permutation matrix $N_{\ell,b} \in \{0,1\}^5$, its ring-embedding $\mathbf{N}_{\ell,b}$ is defined as a $5m \times 5m$ matrix over the ring R of the form:

$$\mathbf{N}_{\ell,b} = \begin{bmatrix} \mathbf{N}_{\ell,b,1,1} & \ldots & \mathbf{N}_{\ell,b,1,5} \\ \vdots & \ddots & \vdots \\ \mathbf{N}_{\ell,b,5,1} & \ldots & \mathbf{N}_{\ell,b,5,5} \end{bmatrix},$$

where for each $w, v \in [5]$, $\mathbf{N}_{\ell,b,w,v} \in R^{m\times m}$ is as defined below:

$$\mathbf{N}_{\ell,b,w,v} = \begin{cases} \mathbf{0} & \text{if } N_{\ell,b}[w,v] = 0, \\ \text{uniformly random} & \text{otherwise.} \end{cases}$$

Note that by Theorem 4.2, the set of such ring-embedded permutation matrices is closed over the ring R.

Generating Guard Matrices. We generate a sequence of "guard" matrix-pairs $\{(\mathbf{L}_\ell, \mathbf{R}_\ell)\}_{\ell\in[L]}$ (over the ring R underlying the homomorphic synthesizer) such that the following conditions hold:

- For each $\ell \in [L]$, the "left guard" $\mathbf{L}_\ell$ is of dimension $c_1 cm \times c_2 m$ and the "right guard" $\mathbf{R}_\ell$ is of dimension $c_2 m \times c_1 m$ where c_1, c_2 are constants such that $c_1 >> c_2$. Intuitively, each left guard will be very "tall and skinny" matrix, while each right guard is a "short and fat matrix".

- For each $\ell \in [L-1]$, we have
$$\mathbf{R}_\ell \mathbf{L}_{\ell+1} = \mathbf{D}_\ell \in R^{c_2 m \times c_2 m},$$
where $\mathbf{D}_\ell$ is a "block-diagonal" matrix of the form
$$\mathbf{D}_\ell = \begin{bmatrix} \mathbf{D}_{\ell,1} & & & & & \\ & \mathbf{D}_{\ell,2} & & \mathbf{0}_R & & \\ & & \mathbf{D}_{\ell,3} & & & \\ & & & \ddots & & \\ & \mathbf{0}_R & & & \ddots & \\ & & & & & \mathbf{D}_{\ell,c_2 m} \end{bmatrix},$$
where for each $j \in [c_2 m]$, $\mathbf{D}_{\ell,j}$ is a square matrix of dimension $m \times m$ over the ring R. More formally, suppose that for some $\ell \in [L-1]$, the guard matrices $\mathbf{R}_\ell$ and $\mathbf{L}_{\ell+1}$ have the following structure:
$$\mathbf{R}_\ell = \begin{bmatrix} -- & -- & \mathbf{R}_{\ell,1} & -- & -- \\ & & \vdots & & \\ -- & -- & \mathbf{R}_{\ell,c_2 m} & -- & -- \end{bmatrix}, \quad \mathbf{L}_{\ell+1} = \begin{bmatrix} | & & | \\ | & & | \\ \mathbf{L}_{\ell+1,1} & \cdots & \mathbf{L}_{\ell+1,c_2 m} \\ | & & | \\ | & & | \end{bmatrix}.$$

Then for each $j, j' \in [c_2 m]$, we have
$$\mathbf{R}_{\ell,j}\mathbf{L}_{\ell,j'} = \begin{cases} \mathbf{D}_{\ell,j} & \text{if } j = j', \\ \mathbf{0} & \text{if } j \neq j'. \end{cases}$$

We now show that such a sequence of matrix-pairs can be created efficiently. For each $\ell \in [L-1]$ and parameter m as described above, do the following:

1. Sample uniform matrices $\mathbf{A}_\ell \leftarrow R^{c_2 m \times (c_1/2 - c_2)m}$ and $\mathbf{B}_\ell \leftarrow R^{c_2 m \times (c_1/2 - c_2)m}$, and set
$$\mathbf{Y}_\ell = \begin{bmatrix} \mathbf{A}_\ell & \mathbf{J} \end{bmatrix}, \quad \mathbf{Z}_\ell = \begin{bmatrix} \mathbf{B}_\ell \\ \mathbf{J}^{-1}(\mathbf{D} - \mathbf{A}_\ell \mathbf{B}_\ell) \end{bmatrix},$$
where $\mathbf{D}_\ell$ is a uniformly sampled "block-diagonal" matrix $R^{c_2 m \times c_2 m}$ as described above, and $\mathbf{J}$ is an *upper triangular* matrix in $R^{c_2 m \times c_2 m}$ with an efficiently computable inverse. More specifically, we have:
$$\mathbf{J} = \begin{bmatrix} 1_R & 1_R & 1_R & \dots & 1_R \\ & 1_R & 1_R & \dots & 1_R \\ & & \ddots & \ddots & \vdots \\ & \mathbf{0}_R & & \ddots & 1_R \\ & & & & 1_R \end{bmatrix}, \quad \mathbf{J}^{-1} = \begin{bmatrix} 1_R & (-1)_R & 0_R & 0_R & \dots & 0_R \\ & 1_R & (-1)_R & 0_R & \dots & 0_R \\ & & \ddots & \ddots & \ddots & \vdots \\ & & & \ddots & \ddots & 0_R \\ & \mathbf{0}_R & & & \ddots & (-1)_R \\ & & & & & 1_R \end{bmatrix},$$
where $(-1)_R$ denotes the additive inverse of 1_R over the ring R and $\mathbf{0}_R$ denotes an all-zero sub-matrix of appropriate dimensions.

2. Sample a uniform square matrix $\mathbf{X}_\ell \leftarrow R^{(c_1/2)m \times (c_1/2)m}$ and a uniform matrix $\mathbf{U}_\ell \leftarrow R^{c_2 m \times (c_1/2)m}$, and set $\mathbf{R}_\ell$ and $\mathbf{L}_{\ell+1}$ as:
$$\mathbf{R}_\ell = \begin{bmatrix} \mathbf{U}_\ell & \quad \mathbf{U}_\ell \mathbf{X}_\ell + \mathbf{Y}_\ell \end{bmatrix}, \quad \mathbf{L}_{\ell+1} = \begin{bmatrix} -\mathbf{X}_\ell \mathbf{Z}_\ell \\ \\ \mathbf{Z}_\ell \end{bmatrix}$$

It easy to see that for each $\ell \in [L-1]$, we have

$$\mathbf{R}_\ell \mathbf{L}_{\ell+1} = -\mathbf{U}_\ell \mathbf{X}_\ell \mathbf{Z}_\ell + (\mathbf{U}_\ell \mathbf{X}_\ell + \mathbf{Y}_\ell)\, \mathbf{Z}_\ell = \mathbf{Y}_\ell \mathbf{Z}_\ell = \mathbf{A}_\ell \mathbf{B}_\ell + \mathbf{D}\mathbf{D}^{-1} (\mathbf{D} - \mathbf{A}_\ell \mathbf{B}_\ell) = \mathbf{D}.$$

Note that we do not explicitly generate the "end-guard" matrices $\mathbf{L}_1$ and $\mathbf{R}_L$. For the moment, we assume that these guard matrices are set to the identity matrix $\mathbf{I}$ over $R^{m \times m}$.

A Simple iO construction. We are now ready to describe a simple version of our iO construction. Given an oblivious branching program of depth L, the obfuscation algorithm does the following :

1. **Step-1:** Construct a sequence of ring-embedded permutation matrices of the form $\{\mathbf{N}_{\ell,b}\}_{\ell \in [L], b \in \{0,1\}}$ as described above.

2. **Step-2:** Generate a sequence of "guard" matrix-pairs $\{(\mathbf{L}_\ell, \mathbf{R}_\ell)\}_{\ell \in [L]}$ satisfying the constraints as described above, for constants $c_2 = 5$ and $c_1 >> 5$.

3. **Step-3:** Generate a a sequence of $2L$ "guarded" program matrix encodings $\{\widetilde{\mathbf{N}}_{\ell,b}\}_{\ell \in [L], b \in \{0,1\}}$, where
$$\widetilde{\mathbf{N}}_{\ell,b} = \mathbf{L}_\ell \mathbf{N}_{\ell,b} \mathbf{R}_\ell.$$

Evaluation and Zero-testing. To efficiently evaluate the program on an input $\mathbf{x} = (x_1, \ldots, x_N)$, one can compute the following "subset-product" of the "guarded" program matrix encodings:

$$\mathbf{Q} = \prod_{\ell=1}^{L} \widetilde{\mathbf{N}}_{\ell, x_{\phi(\ell)}}.$$

It is easy to see that the final product $\mathbf{Q}$ is matrix of dimension $5m \times 5m$. Suppose, $\mathbf{Q}$ is structured as follows:

$$\mathbf{Q} = \begin{bmatrix} \mathbf{Q}_{1,1} & \cdots & \mathbf{Q}_{1,5} \\ \vdots & \ddots & \vdots \\ \mathbf{Q}_{5,1} & \cdots & \mathbf{Q}_{5,5} \end{bmatrix},$$

Also, let Q be the permutation matrix resulting from performing the same "subset-product" on the actual permutation matrices in the clear, i.e., let

$$Q = \prod_{\ell=1}^{L} \mathbf{M}_{\ell, x_{\phi(\ell)}}.$$

The zero test procedure crucially uses the following relation that holds with overwhelmingly large probability between each submatrix $\mathbf{Q}_{w,v}$ and the permutation matrix Q for any $(w, v) \in [5] \times [5]$:

$$\mathbf{Q}_{w,v} = \mathbf{0} \ \text{ if and only if } Q[w, v] = 0.$$

The zero test will then choose a single non-diagonal entry (i, j). Note that this entry is non-zero whenever the branching program outputs 0, i.e., whenever the matrix product Q is not the identity permutation matrix. Hence, it suffices for the zero test to check whether the corresponding submatrix $\mathbf{Q}_{i,j}$ is zero or non-zero.

This construction gives us the desired functionality. However, it is not secure since it does not ensure consistency.

Enforcing Consistency. We now augment the aforementioned construction to enforce consistency. In other words, for a variable x_i that is associated with multiple levels of the program, the check should enforce that the same value of x_i is used at all the associated levels. We handle this in our construction by making two main alterations to the previous construction:

1. We increase the number of matrix encodings used for the construction from $2L$ to $2L + 2N$. We refer to the first set of $2L$ block diagonal matrices as "program-carrying matrices" and the next set of $2N$ block diagonal matrices as "enforcer matrices", following the nomenclature introduced in [GLSW15].
2. In addition to the ring-embedded program matrices, we incorporate "consistency sub-matrices" into both sets of encodings.

We now describe in details how each of these steps are executed:

1. **Generating Program-Carrying Matrices.** Recall that in the previous construction, at each level $\ell \in [L]$ and for each bit $b \in \{0,1\}$, we had a ring-embedded permutation matrix $\mathbf{N}_{\ell,b}$ of dimension $5m \times 5m$, structured as follows:
$$\mathbf{N}_{\ell,b} = \begin{bmatrix} \mathbf{N}_{\ell,b,1,1} & \cdots & \mathbf{N}_{\ell,b,1,5} \\ \vdots & \ddots & \vdots \\ \mathbf{N}_{\ell,b,5,1} & \cdots & \mathbf{N}_{\ell,b,5,5} \end{bmatrix},$$
For each level $\ell \in [L]$ and for each bit $b \in \{0,1\}$, we now construct a set of $5 \cdot 5$ block diagonal 'program-carrying matrices" $\{\mathbf{P}_{\ell,b,w,v}\}_{w,v \in [5]}$, each of dimension $(2(L+N)+1)m \times (2(L+N)+1)m$, structured as:
$$\mathbf{P}_{\ell,b,w,v} = \begin{bmatrix} \mathbf{N}_{\ell,b,w,v} & & & & & \\ & \mathbf{C}_{(\ell,1),(b,0),(w,v)} & & & & \\ & & \mathbf{C}_{(\ell,1),(b,1),(w,v)} & & & \\ & & & \ddots & & \\ & & & & \mathbf{C}_{(\ell,L+N),(b,0),(w,v)} & \\ & & & & & \mathbf{C}_{(\ell,L+N),(b,1),(w,v)} \end{bmatrix},$$
where for each $\ell \in [L]$, each $\ell' \in [L+N]$, each $b, b' \in \{0,1\}$ and each matrix position $(w,v) \in [5] \times [5]$, we have
$$\mathbf{C}_{(\ell,\ell'),(b,b'),(w,v)} = \begin{cases} \mathbf{0} & \text{if } (\ell', b') = (\ell, 1-b), \\ \text{uniform in } R^{m \times m} & \text{otherwise.} \end{cases}.$$
2. **Generating Enforcer Matrices.** Next, for each variable index $i \in [N]$ and for each bit $b \in \{0,1\}$, we construct an additional block diagonal 'enforcer matrices" $\mathbf{E}_{i,b}$ of dimension $(2(L+N)+1)m \times (2(L+N)+1)m$, structured as:
$$\mathbf{E}_{i,b} = \begin{bmatrix} \mathbf{T}_{i,b} & & & & & \\ & \mathbf{C}_{(i,1),(b,0)} & & & & \\ & & \mathbf{C}_{(i,1),(b,1)} & & & \\ & & & \ddots & & \\ & & & & \mathbf{C}_{(i,L+N),(b,0)} & \\ & & & & & \mathbf{C}_{(i,L+N),(b,1)} \end{bmatrix},$$
where for each $i \in [N]$, each $\ell' \in [L+N]$, and each $b, b' \in \{0,1\}$, we have $\mathbf{T}_{i,b} \leftarrow R^{m \times m}$, and
$$\mathbf{C}_{(\ell,\ell'),(b,b')} = \begin{cases} \mathbf{0} & \text{if } (i, b') = (\phi\ell, b), \\ \text{uniform in } R^{m \times m} & \text{otherwise.} \end{cases},$$
where recall that $\phi : [L] \to [N]$ is a mapping such that for each $\ell \in [L]$, $i = \phi(\ell)$ indicates which variable x_i controls the ℓ^{th} level branch.

3. **Guarding Program-Carrying And Enforcer Matrices.** As before, we generate a sequence of "guard" matrix-pairs. Notice that we now need $2(L+N)$ guard matrices in order to cover both the program-carrying and enforcer matrices. More specifically, we generate a sequence of guard matrices $\{(\mathbf{L}_\ell, \mathbf{R}_\ell)\}_{\ell\in[L+N]}$ satisfying the constraints as described above, albeit for constants $c_2 = 2(L+N)+1$ and $c_1 >> 2(L+N)+1$.

   Note that one difference from the simple iO construction presented earlier is in how we generate the "end-guard" matrices $\mathbf{L}_1$ and $\mathbf{R}_L$. In the simple construction, we assumed that these guard matrices were set to the identity matrix $\mathbf{I}$ over $R^{m\times m}$. For this construction, we assume that $\mathbf{L}_1 \in R^{m\times(2(L+N)+1)m}$ and $\mathbf{R}_L \in R^{(2(L+N)+1)m\times m}$ are structured as follows:

$$\mathbf{L}_1 = \begin{bmatrix}\mathbf{I} & \mathbf{I} & \dots & \mathbf{I}\end{bmatrix}, \quad \mathbf{R}_L = \begin{bmatrix}\mathbf{I}\\ \mathbf{I}\\ \vdots \\ \mathbf{I}\end{bmatrix}.$$

   Finally, we create a sequence of $(5\cdot 5\cdot 2L + 2N)$ "guarded" program matrix encodings of the form

$$\{\widetilde{\mathbf{P}}_{\ell,b,w,v}\}_{\ell\in[L+N],b\in\{0,1\},w\in[5],v\in[5]}, \quad \{\widetilde{\mathbf{E}}_{i,b}\}_{\ell\in[L+N],b\in\{0,1\}},$$

   where for each level $\ell\in[L]$, each bit $b\in\{0,1\}$ and each matrix position $(w,v)\in[5]\times[5]$, we have

$$\widetilde{\mathbf{N}}_{\ell,b,w,v} = \mathbf{L}_\ell \mathbf{P}_{\ell,b,w,v}\mathbf{R}_\ell,$$

   and for each variable index $i\in[N]$ and for each bit $b\in\{0,1\}$, we have

$$\widetilde{\mathbf{E}}_{i,b} = \mathbf{L}_{L+i}\mathbf{E}_{i,b}\mathbf{R}_{L+i}.$$

Evaluation and Zero-testing. To efficiently evaluate the program on an input $\mathbf{x} = (x_1,\dots,x_N)$, one can compute the following "subset-product" of the "guarded" program matrix encodings:

$$\mathbf{Q} = \prod_{\ell=1}^{L} \widetilde{\mathbf{P}}_{\ell,x_{\phi(\ell)}},$$

where each $\widetilde{\mathbf{P}}_{\ell,x_{\phi(\ell)}}$ may be viewed as a 5×5 super-matrix of the corresponding program-carrying matrices

$$\{\widetilde{\mathbf{P}}_{\ell,x_{\phi(\ell)},w,v}\}_{w,v\in[5]}.$$

Suppose, $\mathbf{Q}$ is structured as follows:

$$\mathbf{Q} = \begin{bmatrix}\mathbf{Q}_{1,1} & \dots & \mathbf{Q}_{1,5}\\ \vdots & \ddots & \vdots\\ \mathbf{Q}_{5,1} & \dots & \mathbf{Q}_{5,5}\end{bmatrix},$$

Also, let Q be the permutation matrix resulting from performing the same "subset-product" on the actual permutation matrices in the clear, i.e., let

$$Q = \prod_{\ell=1}^{L} \mathbf{M}_{\ell,x_{\phi(\ell)}}.$$

As in the simple construction, the zero test will choose a single non-diagonal (i,j) so that it is non-zero as an entry of the subset-product Q whenever the branching program outputs 0 (i.e. whenever the matrix product Q is not the identity permutation matrix). It then performs an additional subset-product of the form

$$\mathbf{Q}' = \mathbf{Q}_{i,j}\prod_{i=1}^{N}\widetilde{\mathbf{E}}_{L+i,x_i}.$$

Observe that $\mathbf{Q}'$ is a matrix of dimension $m\times m$. At this point, the zero test simply checks if $\mathbf{Q}' = \mathbf{0}$. If yes, it outputs 1, else it outputs 0.

Correctness. We first observe that the consistency-check sub-matrices do not contribute to the result. To see this, observe the following:

1. For each (ℓ, b), if $b \neq x_\phi(\ell)$, then the consistency submatrices in the corresponding diagonal slot do not contribute to the final product since there is a zero submatrix in this diagonal slot in the program carrying matrix $\mathbf{P}_{\ell, x_{\phi(\ell)}, w, v}$ for every $(w, v) \in [5] \times [5]$.

2. For each (ℓ, b), if $b = x_\phi(\ell)$, then the consistency submatrices in the corresponding diagonal slot do not contribute to the final product since there is a zero submatrix in this diagonal slot in the enforcer matrix $\mathbf{E}_{L+\phi(\ell), x_{\phi(\ell)}}$.

Thus the only potential contributions to the value of $\mathbf{Q}'$ come from the diagonal slots corresponding to the ring-embedded permutation matrices. It is now easy to see that $\mathbf{Q}' \neq \mathbf{0}$ if and only if the plaintext matrix subset-product Q has a non-zero entry in the position (i, j), indicating that the program P outputs 0 on input $\mathbf{x}$.

5.2 Parallel iO Construction

The next challenge towards our eventual goal of building iaiO is to be able to evaluate multiple oblivious branching programs in parallel on the same set of inputs (where each program has the same size and the same level-to-input mapping, but potentially differs in the contents of their matrices). For the parallel iO construction, we borrow from ideas presented in [GLSW15].

In [GLSW15], the authors present a strategy for executing multiple functionally equivalent programs in parallel by embedding the corresponding permutation matrices from different programs in different subgroups of a multilinear map and "aggregating" them into a single group element. In this section, we present a construction strategy that ports their ideas into the setting of ring-homomorphic synthesizers.

At a high level, we create "aggregate" matrices, which are block-diagonal matrices that have ring-embedded permutation matrices from different programs in different diagonal "slots". The evaluation process remains the same as in the core iO construction, namely, given an input, we compute the corresponding subset product of the aggregate matrices depending on the input.

Recall that each aggregate matrix places permutation matrices from different programs in *different* diagonal "slots". This implies that any subset product of these aggregate matrices would be a block diagonal matrix, where each diagonal "slot" contains the subset product of permutation matrices from the corresponding program. In other words, evaluation results in an "aggregate evaluation matrix", where each diagonal slot contains the evaluation of the corresponding program on the same input.

Note that we ignore input-activations at the moment. In other words, we assume that every input activates every program. It turns out that once we achieve an iO construction capable of evaluating multiple programs in parallel, incorporating input activations into it follows immediately via a few simple tweaks.

Generating Aggregate Matrices. Let $P_1, \ldots, P_T$ be the oblivious branching programs of depth L that are to be handled in parallel, where for each $t \in [T]$, we have

$$P_t = \{M_{t,\ell,0}, M_{t,\ell,1}\}_{\ell \in [L]}.$$

As before, we uniformly randomly sample *additional permutation matrices* $Z_1, \ldots, Z_{L-1} \in \{0,1\}^{5\times 5}$, and define a new set of matrices $\{N_{t,\ell,b}\}_{t\in[T],\ell\in[L],b\in\{0,1\}}$, where

$$N_{t,\ell,b} = Z_{\ell-1}^{-1} M_{t,\ell,b} Z_\ell,$$

where Z_0 and Z_L are both set to be the identity permutation.

Let the corresponding set of ring-embedded permutation matrices be denoted as $\{\mathbf{N}_{t,\ell,b}\}_{t\in[T],\ell\in[L],b\in\{0,1\}}$, where each $\mathbf{N}_{t,\ell,b}$ is structured as follows:

$$\mathbf{N}_{t,\ell,b} = \begin{bmatrix} \mathbf{N}_{t,\ell,b,1,1} & \cdots & \mathbf{N}_{t,ell,b,1,5} \\ \vdots & \ddots & \vdots \\ \mathbf{N}_{t,\ell,b,5,1} & \cdots & \mathbf{N}_{t,\ell,b,5,5} \end{bmatrix}.$$

For each program index $t \in [T]$, each level $\ell \in [L]$ and each bit $b \in \{0,1\}$, we construct an "aggregate" block-diagonal matrix $\mathbf{A}_{\ell,b,w,v}$, structured as:

$$\mathbf{A}_{t,\ell,b} = \begin{bmatrix} \mathbf{N}_{1,\ell,b,w,v} & & \\ & \ddots & \\ & & \mathbf{N}_{T,\ell,b,w,v} \end{bmatrix}.$$

Generating Program-Carrying Matrices. As in the core iO construction, for each level $\ell \in [L]$ and for each bit $b \in \{0,1\}$, we now construct a set of $5 \cdot 5$ block diagonal 'program-carrying matrices" $\{\mathbf{P}'_{\ell,b,w,v}\}_{w,v\in[5]}$, each of dimension $(2(L+N)+T)m \times (2(L+N)+1)m$, structured as:

$$\mathbf{P}'_{\ell,b,w,v} = \begin{bmatrix} \mathbf{A}_{\ell,b,w,v} & & & & & \\ & \mathbf{C}_{(\ell,1),(b,0),(w,v)} & & & & \\ & & \mathbf{C}_{(\ell,1),(b,1),(w,v)} & & & \\ & & & \ddots & & \\ & & & & \mathbf{C}_{(\ell,L+N),(b,0),(w,v)} & \\ & & & & & \mathbf{C}_{(\ell,L+N),(b,1),(w,v)} \end{bmatrix},$$

where, as in the core iO construction, for each $\ell \in [L]$, each $\ell' \in [L+N]$, each $b, b' \in \{0,1\}$ and each matrix position $(w,v) \in [5] \times [5]$, we have

$$\mathbf{C}_{(\ell,\ell'),(b,b'),(w,v)} = \begin{cases} \mathbf{0} & \text{if } (\ell',b') = (\ell,1-b), \\ \text{uniform in } R^{m\times m} & \text{otherwise.} \end{cases}.$$

Note that the only change from the core iO construction is that we now place the "aggregate matrix" $\mathbf{A}_{\ell,b,w,v}$ at the top left corner of each program-carrying matrix as opposed to just a single ring-embedded permutation matrix. As a sanity check, observe that for the special case where the number of programs $T = 1$, a program-carrying matrix in the parallel iO construction becomes identical to that in the core iO construction.

Generating Enforcer Matrices. Next, for each variable index $i \in [N]$ and for each bit $b \in \{0,1\}$, we construct an additional "enforcer matrix" $\mathbf{E}'_{i,b}$ of dimension $(2(L+N)+T)m \times (2(L+N)+T)m$, structured as:

$$\mathbf{E}'_{i,b} = \begin{bmatrix} \mathbf{T}_{i,b} & & & & & \\ & \mathbf{C}_{(i,1),(b,0)} & & & & \\ & & \mathbf{C}_{(i,1),(b,1)} & & & \\ & & & \ddots & & \\ & & & & \mathbf{C}_{(i,L+N),(b,0)} & \\ & & & & & \mathbf{C}_{(i,L+N),(b,1)} \end{bmatrix},$$

where, as in the core iO construction, for each $i \in [N]$, each $\ell' \in [L+N]$, and each $b, b' \in \{0,1\}$, we have

$$\mathbf{C}_{(\ell,\ell'),(b,b')} = \begin{cases} \mathbf{0} & \text{if } (i,b') = (\phi(\ell),b), \\ \text{uniform in } R^{m\times m} & \text{otherwise.} \end{cases},$$

where recall that $\phi : [L] \to [N]$ is a mapping such that for each $\ell \in [L]$, $i = \phi(\ell)$ indicates which variable x_i controls the ℓ^{th} level branch.

The only change from the core iO construction is in how we generate the matrix $\mathbf{T}_{i,b}$. Note that in the core iO construction, we have $\mathbf{T}_{i,b}$ sampled uniformly from $R^{m\times m}$. In the parallel version of the construction, we instead structure $\mathbf{T}_{i,b}$ as a block diagonal matrix in $R^{Tm\times Tm}$, as described below:

$$\mathbf{T}_{i,b} = \begin{bmatrix} \mathbf{T}_{1,i,b} & & \\ & \ddots & \\ & & \mathbf{T}_{T,i,b} \end{bmatrix},$$

where for each program index $t \in [T]$, the block matrix $\mathbf{T}_{t,i,b}$ is sampled uniformly from $R^{m \times m}$. Once again, as a sanity check, observe that for the special case where the number of programs $T = 1$, an enforcer matrix in the parallel iO construction becomes identical to that in the core iO construction.

Generating Guarded Encodings. As in the core iO construction, we generate a sequence of guard matrices $\{(\mathbf{L}_\ell, \mathbf{R}_\ell)\}_{\ell \in [L+N]}$ satisfying the constraints as described above, albeit for constants $c_2 = 2(L+N)+T$ and $c_1 >> 2(L+N)+T$. In particular, we generate "end-guard" matrices $\mathbf{L}_1 \in R^{m \times (2(L+N)+T)m}$ and $\mathbf{R}_L \in R^{(2(L+N)+T)m \times m}$ are structured as follows:

$$\mathbf{L}_1 = \begin{bmatrix} \mathbf{I} & \mathbf{I} & \dots & \mathbf{I} \end{bmatrix}, \quad \mathbf{R}_L = \begin{bmatrix} \mathbf{I} \\ \mathbf{I} \\ \vdots \\ \mathbf{I} \end{bmatrix}.$$

Next, we create a sequence of $(5 \cdot 5 \cdot 2L + 2N)$ "guarded" program matrix encodings of the form

$$\{\widetilde{\mathbf{P}}_{\ell,b,w,v}\}_{\ell \in [L+N], b \in \{0,1\}, w \in [5], v \in [5]}, \quad \{\widetilde{\mathbf{E}}_{i,b}\}_{i \in [N], b \in \{0,1\}},$$

where for each level $\ell \in [L]$, each bit $b \in \{0,1\}$ and each matrix position $(w,v) \in [5] \times [5]$, we have

$$\widetilde{\mathbf{P}}_{\ell,b,w,v} = \mathbf{L}_\ell \mathbf{P}'_{\ell,b,w,v} \mathbf{R}_\ell,$$

and for each variable index $i \in [N]$ and for each bit $b \in \{0,1\}$, we have

$$\widetilde{\mathbf{E}}_{i,b} = \mathbf{L}_{L+i} \mathbf{E}'_{i,b} \mathbf{R}_{L+i}.$$

Evaluation and Zero-testing. To efficiently evaluate the program on an input $\mathbf{x} = (x_1, \dots, x_N)$, one can compute the following "subset-product" of the "guarded" program matrix encodings:

$$\mathbf{Q} = \prod_{\ell=1}^{L} \widetilde{\mathbf{P}}_{\ell, x_{\phi(\ell)}},$$

where each $\widetilde{\mathbf{P}}_{\ell, x_{\phi(\ell)}}$ may be viewed as a 5×5 super-matrix of the corresponding program-carrying matrices

$$\{\widetilde{\mathbf{P}}_{\ell, x_{\phi(\ell)}, w, v}\}_{w,v \in [5]}.$$

Suppose, $\mathbf{Q}$ is structured as follows:

$$\mathbf{Q} = \begin{bmatrix} \mathbf{Q}_{1,1} & \dots & \mathbf{Q}_{1,5} \\ \vdots & \ddots & \vdots \\ \mathbf{Q}_{5,1} & \dots & \mathbf{Q}_{5,5} \end{bmatrix},$$

Also, let Q_t be the permutation matrix resulting from performing same "subset-product" on the actual permutation matrices of program P_t in the clear, i.e., let

$$Q_t = \prod_{\ell=1}^{L} \mathbf{M}_{t,\ell,x_{\phi(\ell)}}.$$

As in the simple construction, the zero test will choose a single non-diagonal (i,j) so that it is non-zero as an entry of the subset-product Q whenever the branching program outputs 0 (i.e. whenever the matrix product Q is not the identity permutation matrix). It then performs an additional subset-product of the form

$$\mathbf{Q}' = \mathbf{Q}_{i,j} \prod_{i=1}^{N} \widetilde{\mathbf{E}}_{L+i, x_i}.$$

Observe that $\mathbf{Q}'$ is a matrix of dimension $m \times m$. At this point, the zero test simply checks if $\mathbf{Q}' = \mathbf{0}$. If yes, it outputs 1, else it outputs 0.

Correctness. We first observe that the consistency-check sub-matrices do not contribute to the result. To see this, observe the following:

1. For each (ℓ, b), if $b \neq x_\phi(\ell)$, then the consistency submatrices in the corresponding diagonal slot do not contribute to the final product since there is a zero submatrix in this diagonal slot in the program carrying matrix $\mathbf{P}_{\ell, x_{\phi(\ell)}, w, v}$ for every $(w, v) \in [5] \times [5]$.

2. For each (ℓ, b), if $b = x_\phi(\ell)$, then the consistency submatrices in the corresponding diagonal slot do not contribute to the final product since there is a zero submatrix in this diagonal slot in the enforcer matrix $\mathbf{E}_{L+\phi(\ell), x_{\phi(\ell)}}$.

Thus the only potential contributions to the value of $\mathbf{Q}'$ come from the diagonal slots corresponding to the ring-embedded permutation matrices. It is now easy to see that $\mathbf{Q}' \neq \mathbf{0}$ if and only if the plaintext matrix subset-product Q_t corresponding to some program P_t has a non-zero entry in the position (i, j), indicating that this program P_t outputs 0 on input $\mathbf{x}$.

5.3 Incorporating Input-Activations: iaiO Construction

The previous iO construction may be viewed as an iaiO construction **with the all-ones input activation matrix**, i.e., where every input activates every program. In order to achieve full-fledged iaiO, we need to incorporate arbitrary input activations into this construction. We do this by tweaking certain parts of the "enforcer matrices" in the previous iO construction. The tweaks are based on ideas presented in the [GLSW15] paper.

Recall that in the parallel iO construction, for each variable index $i \in [N]$ and for each bit $b \in \{0,1\}$, we generated a block diagonal 'enforcer matrix" $\mathbf{E}'_{i,b}$ of dimension $(2(L+N)+T)m \times (2(L+N)+T)m$, structured as:

$$\mathbf{E}'_{i,b} = \begin{bmatrix} \mathbf{T}_{i,b} & & & & & \\ & \mathbf{C}_{(i,1),(b,0)} & & & & \\ & & \mathbf{C}_{(i,1),(b,1)} & & & \\ & & & \ddots & & \\ & & & & \mathbf{C}_{(i,L+N),(b,0)} & \\ & & & & & \mathbf{C}_{(i,L+N),(b,1)} \end{bmatrix},$$

where, we structured $\mathbf{T}_{i,b}$ as a block diagonal matrix in $R^{Tm \times Tm}$, as described below:

$$\mathbf{T}_{i,b} = \begin{bmatrix} \mathbf{T}_{1,i,b} & & \\ & \dots & \\ & & \mathbf{T}_{T,i,b} \end{bmatrix},$$

where for each program index $t \in [T]$, the block matrix $\mathbf{T}_{t,i,b}$ was sampled uniformly from $R^{m \times m}$.

In the full-fledged iaiO construction, we generate these enforcer matrices in exactly the same way, except for the manner in which each $\mathbf{T}_{t,i,b,w,v}$ block matrix is generated. We generate each $\mathbf{T}_{t,i,b}$ block matrix depending on some additional input activation information that the obfuscation algorithm takes as input.

More concretely, let $\mathbf{G}$ be the "activation matrix" of dimension $N \times T \times 2$, where each $i \in [N]$ is the "row" corresponding to the input bits, each $t \in [T]$ is the "column" corresponding to the program number, and $b \in \{0,1\}$ is the bit corresponding to activation on the particular input bit value. Now, for each $i \in [N]$, $t \in [T]$ and $b \in \{0,1\}$, we generate the matrix $\mathbf{T}_{t,i,b}$ as follows:

$$\mathbf{T}_{t,i,b} = \begin{cases} \mathbf{0} & \text{if } \mathbf{G}[i,t,b] = 0, \\ \text{uniform in } R^{m \times m} & \text{otherwise.} \end{cases}.$$

Evaluation and Zero-testing. Evaluation and zero-testing for the iaiO construction is exactly the same as in the parallel iO construction. Nonetheless, we repeat it here for the sake of completeness.

To efficiently evaluate the program on an input $\mathbf{x} = (x_1, \ldots, x_N)$, one can compute the following "subset-product" of the "guarded" program matrix encodings:

$$\mathbf{Q} = \prod_{\ell=1}^{L} \widetilde{\mathbf{P}}_{\ell, x_{\phi(\ell)}},$$

where each $\widetilde{\mathbf{P}}_{\ell, x_{\phi(\ell)}}$ may be viewed as a 5×5 super-matrix of the corresponding program-carrying matrices

$$\{\widetilde{\mathbf{P}}_{\ell, x_{\phi(\ell)}, w, v}\}_{w,v \in [5]}.$$

Suppose, $\mathbf{Q}$ is structured as follows:

$$\mathbf{Q} = \begin{bmatrix} \mathbf{Q}_{1,1} & \cdots & \mathbf{Q}_{1,5} \\ \vdots & \ddots & \vdots \\ \mathbf{Q}_{5,1} & \cdots & \mathbf{Q}_{5,5} \end{bmatrix},$$

Also, let Q_t be the permutation matrix resulting from performing same "subset-product" on the actual permutation matrices of program P_t in the clear, i.e., let

$$Q_t = \prod_{\ell=1}^{L} \mathbf{M}_{t, \ell, x_{\phi(\ell)}}.$$

As in the parallel iO construction, the zero test will choose a single non-diagonal (i, j) so that it is non-zero as an entry of the subset-product Q whenever the branching program outputs 0 (i.e. whenever the matrix product Q is not the identity permutation matrix). It then performs an additional subset-product of the form

$$\mathbf{Q}' = \mathbf{Q}_{i,j} \prod_{i=1}^{N} \widetilde{\mathbf{E}}_{L+i, x_i}.$$

Observe that $\mathbf{Q}'$ is a matrix of dimension $m \times m$. At this point, the zero test simply checks if $\mathbf{Q}' = \mathbf{0}$. If yes, it outputs 1, else it outputs 0.

Correctness. We first observe that the consistency-check sub-matrices do not contribute to the result. To see this, observe the following:

1. For each (ℓ, b), if $b \neq x_\phi(\ell)$, then the consistency submatrices in the corresponding diagonal slot do not contribute to the final product since there is a zero submatrix in this diagonal slot in the program carrying matrix $\mathbf{P}_{\ell, x_{\phi(\ell)}, w, v}$ for every $(w, v) \in [5] \times [5]$.

2. For each (ℓ, b), if $b = x_\phi(\ell)$, then the consistency submatrices in the corresponding diagonal slot do not contribute to the final product since there is a zero submatrix in this diagonal slot in the enforcer matrix $\mathbf{E}_{L+\phi(\ell), x_{\phi(\ell)}}$.

Thus the only potential contributions to the value of $\mathbf{Q}'$ come from the diagonal slots corresponding to the ring-embedded permutation matrices.

Additionally, observe the following:

1. Suppose some program P_t *is not* activated by the input $\mathbf{x}$. Then, the input-activation matrix $\mathbf{G}$ has a 0 in the slot corresponding to x_i on some row $i \in [N]$ and in column t. Then one of the included enforcing matrices will prevent the program P_t from contributing to the final output.

2. Suppose instead that the program P_t *is* activated by the input $\mathbf{x}$. Then, the input-activation matrix $\mathbf{G}$ has a 1 in the slot corresponding to x_i on some row $i \in [N]$ and in column t. Hence, none of the included enforcing matrices will prevent the program P_t from contributing to the final output.

It is now easy to see that $\mathbf{Q}' \neq \mathbf{0}$ if and only if the plaintext matrix subset-product Q_t corresponding to some *activated* program P_t has a non-zero entry in the position (i, j), indicating that this program P_t outputs 0 on input $\mathbf{x}$.

6 The Subspace Hiding Assumption

In this section we define our subspace hiding assumption. It can be viewed as analogous to the assumption of [GLSW15] in spirit, although it is syntactically very different. Before formally describing the assumption, we introduce certain notations used in describing the assumptions.

Diagonal Matrices. Consider a class of block diagonal matrices $\mathbf{D} \in R^{(n\ell)\times(n\ell)}$. We break down $\mathbf{D}$ into an ℓ by ℓ blockwise structure: in other words, we assume the $\mathbf{D}$ is composed of ℓ^2 square blocks of size $n \times n$, which we denote $\mathbf{D}_{i,j}$, and we assume that $\mathbf{D}_{i,j} = 0$ if $i \neq j$. We generally drop the double subscript and represent the diagonal entries by $\mathbf{D}_i$. Graphically, this looks like

$$\mathbf{D} = \begin{bmatrix} \mathbf{D}_1 & & & & \\ & \mathbf{D}_2 & & \text{\Large 0}_R & \\ & & \mathbf{D}_3 & & \\ & & & \ddots & \\ & \text{\Large 0}_R & & & \ddots & \\ & & & & & \mathbf{D}_\ell \end{bmatrix},$$

We introduce some more notation around diagonal matrices which will allow us to simplify our presentation in this section:

1. $\mathbf{D}^i \in R^{(n\ell)\times(n\ell)}$ denotes a block diagonal matrix with the restriction that the i^{th} diagonal block is zero.

2. $\mathbf{D}^{i,j} \in R^{(n\ell)\times(n\ell)}$ denote a block diagonal matrix with the restriction that the i^{th} and j^{th} diagonal blocks are zero.

3. $\widetilde{\mathbf{D}^i} \in R^{(n\ell)\times(n\ell)}$ denotes a block diagonal matrix with the restriction that *all diagonal blocks except the* i^{th} *diagonal block* are zero.

4. $\widetilde{\mathbf{D}^{i,j}} \in R^{(n\ell)\times(n\ell)}$ denotes a block diagonal matrix with the restriction that *all diagonal blocks except the* i^{th} *and* j^{th} *diagonal blocks* are zero.

We will sometimes need to compress such diagonal matrices into rows and columns. We let $\mathbf{I}^\ell$ denote the matrix consisting of a tensor of the identity matrix in n dimensions with the ℓ-dimensional all 1_Rs vector. Correspondingly, we let $(\mathbf{I}^\ell)^T$ be the transpose of $\mathbf{I}^\ell$. We use these matrices to extract the non-zero block diagonals of our diagonal matrices into rows and columns, respectively.

The Guard Matrices. Let n, m and ℓ be integers such that $n\ell << m$. We use a set of matrices $\mathbf{L}_i \in R^{m\times(n\ell)}$ for $i \in [2, z]$, and $\mathbf{R}_i \in R^{(n\ell)\times m}$ for $i \in [1, z-1]$. In addition, we consider submatrices of $\mathbf{L}_i$ and $\mathbf{R}_i$, respectively. Let $\mathbf{L}_{i,j} \in R^{m\times n}$ be the submatrix of $\mathbf{L}_i$ consisting of the $(j-1)\,n + 1$th through the jnth *columns* of $\mathbf{L}_i$, and let $\mathbf{R}_{i,j} \in R^{n\times m}$ be the submatrix of $\mathbf{R}_i$ consisting of the $(j-1)\,n + 1$th through the jnth *rows* of $\mathbf{R}_i$, and let $\mathbf{R}_{i,j}$. In other words, our matrices have the following structure:

$$\mathbf{R}_i = \begin{bmatrix} -- & -- & \mathbf{R}_{i,1} & -- & -- \\ -- & -- & \mathbf{R}_{i,2} & -- & -- \\ & & \vdots & & \\ -- & -- & \mathbf{R}_{i,\ell-1} & -- & -- \\ -- & -- & \mathbf{R}_{i,\ell} & -- & -- \end{bmatrix}, \qquad \mathbf{L}_i = \begin{bmatrix} | & | & & | & | \\ | & | & & | & | \\ \mathbf{L}_{i,1} & \mathbf{L}_{i,2} & \cdots & \mathbf{L}_{i,\ell-1} & \mathbf{L}_{1,\ell} \\ | & | & & | & | \\ | & | & & | & | \end{bmatrix}$$

Experiment $\mathsf{Expt}^{\mathsf{Subspace\text{-}Hiding}}_{R,\ell,z,b}$:

1. The adversary $\mathcal{A}$ chooses two challenge-slots $i_0^*, i_1^* \in [\ell]$ and a challenge-index $j^* \in [z]$. It provides (i_0^*, i_1^*, j^*) to the challenger.
2. The challenger provides the adversary with the following:
   (a) **Generators:** For each $j \in [z]$ and each $i \in [\ell] \setminus \{i_1^*\}$: many samples of the form $\mathbf{L}_j\widetilde{\mathbf{D}^i}\mathbf{R}_j$, where $\widetilde{\mathbf{D}^i}$ is a randomly sampled matrix of the appropriate form as described above, except for when $j = 1$ and $j = z$, where the samples are of the form $\mathbf{I}^\ell\widetilde{\mathbf{D}^i}\mathbf{R}_1$ and $\mathbf{L}_z\widetilde{\mathbf{D}^i}(\mathbf{I}^\ell)^T$, respectively.
   (b) **Challenge-Relevant Elements**: For each $j \in [z]$: many samples of the form $\mathbf{L}_j\widetilde{\mathbf{D}^{i_0^*,i_1^*}}\mathbf{R}_j$, where $\widetilde{\mathbf{D}^{i_0^*,i_1^*}}$ is a randomly sampled matrix of the appropriate form, except for when $j = 1$ and $j = z$, where the samples are of the form $\mathbf{I}^\ell\widetilde{\mathbf{D}^{i_0^*,i_1^*}}\mathbf{R}_1$ and $\mathbf{L}_z\widetilde{\mathbf{D}^{i_0^*,i_1^*}}(\mathbf{I}^\ell)^T$, respectively.
   (c) **Special Challenge-Index Elements**: Many samples of the form $\mathbf{L}_{j^*}\widetilde{\mathbf{D}^{i_1^*}}\mathbf{R}_{j^*}$, where $\widetilde{\mathbf{D}^{i_1^*}}$ is a randomly sampled matrix of the appropriate form, except for when $j^* = 1$ or $j^* = z$, where the samples are of the form $\mathbf{I}^\ell\widetilde{\mathbf{D}^{i_1^*}}\mathbf{R}_1$ or $\mathbf{L}_z\widetilde{\mathbf{D}^{i_1^*}}(\mathbf{I}^\ell)^T$, respectively.
   (d) **The Challenge Element**: A sample that is of one of the two following forms:
      - $\mathbf{L}_{j^*}\widetilde{\mathbf{D}^{i_0^*}}\mathbf{R}_{j^*}$ if $b = 0$, or
      - $\mathbf{L}_{j^*}\widetilde{\mathbf{D}^{i_0^*,i_1^*}}\mathbf{R}_{j^*}$ if $b = 1$,

      where $\widetilde{\mathbf{D}^{i_0^*}}$ and $\widetilde{\mathbf{D}^{i_0^*,i_1^*}}$ are randomly sampled matrices of the appropriate form, except for when $j^* = 1$, where the sample is of one of the two following forms:
      - $\mathbf{I}^\ell\widetilde{\mathbf{D}^{i_0^*}}\mathbf{R}_1$ if $b = 0$, or
      - $\mathbf{I}^\ell\widetilde{\mathbf{D}^{i_0^*,i_1^*}}\mathbf{R}_1$ if $b = 1$,

      or when $j^* = z$, where the sample is of one of the two following forms:
      - $\mathbf{L}_z\widetilde{\mathbf{D}^{i_0^*}}(\mathbf{I}^\ell)^T$ if $b = 0$, or
      - $\mathbf{L}_z\widetilde{\mathbf{D}^{i_0^*,i_1^*}}(\mathbf{I}^\ell)^T$ if $b = 1$.

Figure 2: The Subspace Hiding Assumption

Intuitively, $\mathbf{R}_i$ will be very "short and fat," and $\mathbf{L}_i$ will be very "tall and skinny."

We additionally require the following properties of the $\mathbf{L}_i$ and $\mathbf{R}_i$ matrices: for every $j \neq j'$, it must be the case that

$$\mathbf{R}_{i,j} \cdot \mathbf{L}_{i+1,j'} = 0_R$$

In other words, note that the product $\mathbf{R}_{i,j} \cdot \mathbf{L}_{i+1}$ is required to have the form of a block diagonal matrix $\mathbf{D}$ as we defined above.

The Subspace Hiding Assumptions. Equipped with the aforementioned notations, we now state the subspace hiding assumption. For any ring R, any choice of parameters ℓ, z and each $b \in \{0, 1\}$, let $\mathsf{Expt}^{\mathsf{Subspace\text{-}Hiding}}_{R,\ell,z,b}$ denote an experiment between a challenger and an adversary $\mathcal{A}$ as defined in Figure 2.

Definition 6.1. (Subspace Hiding Assumption.) The subspace hiding assumption is said to hold over a ring R if for any security parameter λ, any choice of parameters $\ell, z = \mathrm{poly}(\lambda)$ and for any PPT adversary $\mathcal{A}$, the views of the adversary $\mathcal{A}$ in the experiments $\mathsf{Expt}^{\mathsf{Subspace\text{-}Hiding}}_{R,\ell,z,0}$ and $\mathsf{Expt}^{\mathsf{Subspace\text{-}Hiding}}_{R,\ell,z,1}$ are computationally indistinguishable.

We also define a special case of this assumption, called the "baby" subspace hiding assumption, for a fixed choice of parameter $\ell = 2$.

Definition 6.2. (Baby Subspace Hiding Assumption.) The "baby" subspace hiding assumption is said to hold over a ring R if for any security parameter λ, any choice of parameter $z = \mathrm{poly}(\lambda)$ and for any PPT adversary $\mathcal{A}$, the views of the adversary $\mathcal{A}$ in the experiments $\mathsf{Expt}^{\mathsf{Subspace\text{-}Hiding}}_{R,2,z,0}$ and $\mathsf{Expt}^{\mathsf{Subspace\text{-}Hiding}}_{R,2,z,1}$ are computationally indistinguishable.

The next sections are organized as follows:

- In Section 7 we prove the security of our iaiO construction based on the subspace hiding assumption as in Definition 6.1.
- In Section 8, we extend the above result to the the subspace hiding assumption as in Definition 6.1.
- Finally, in Section 9, we prove that the baby subspace hiding assumption as in Definition 6.2 holds over any ring R that is the output ring of a ring-embedded homomorphic synthesizer.

Putting these together, we show that the existence of ring-embedded homomorphic synthesizer implies the existence of a secure iaiO scheme.

7 Proof of Security for Our iaiO Construction

In this section, we prove (a) inter-column security, (b) single-input program switching security, (c) completely inactive program security, and (d) intra-column security of our iaiO construction, all based on the subspace hiding assumption as in Definition 6.1. Refer Section 3 for the formal definitions of these security notions for any iaiO scheme.

7.1 Inter-Column Security

Lemma 7.1. *Assuming that the subspace hiding assumption holds, our iaiO construction achieves inter-column security.*

Proof. Suppose that there exists some PPT attacker $\mathcal{A}$ which achieves non-negligible advantage in the inter-column security game for some valid setting of activation matrix $\mathbf{G}$, challenge program indices $t_0^*, t_1^* \in [T]$, challenge variable index $i^* \in [N]$ and challenge bit $b^* \in \{0, 1\}$. We will create a PPT algorithm $\mathcal{B}$ that achieves a non-negligible advantage in breaking the subspace hiding assumption (parameterized in this case by the challenge-index $j^* = (L + i^*)$ and the challenge-slots $(i_0^*, i_1^*) = (t_0^*, t_1^*)$).

Inputs to $\mathcal{B}$. As described in Section 6, $\mathcal{B}$ receives as input several terms of the following kinds:

- **Generators:** For each $\ell \in [L+N]$ and each $j \in [2(L+N)+T] \setminus \{t_1^*\}$: many samples of the form $\mathbf{L}_\ell \widetilde{\mathbf{D}^j} \mathbf{R}_\ell$, where $\widetilde{\mathbf{D}^j}$ is a randomly sampled matrix of the appropriate form, and the "end guards" $\mathbf{L}_1$ and $\mathbf{R}_{L+N}$ are specially structured, as described in Section 5.2.

- **Challenge-Relevant Elements**: For each $\ell \in [L+N]$: many samples of the form $\mathbf{L}_\ell \widetilde{\mathbf{D}^{t_0^*,t_1^*}} \mathbf{R}_\ell$, where $\widetilde{\mathbf{D}^{t_0^*,t_1^*}}$ is a randomly sampled matrix of the appropriate form, and the "end guards" $\mathbf{L}_1$ and $\mathbf{R}_{L+N}$ are specially structured as mentioned before.

- **Special Challenge-Index Elements**: Many samples of the form $\mathbf{L}_{(L+i^*)} \widetilde{\mathbf{D}^{t_1^*}} \mathbf{R}_{(L+i^*)}$, where $\widetilde{\mathbf{D}^{t_1^*}}$ is a randomly sampled matrix of the appropriate form.

- **Challenge Elements**: A sample that is either of the form $\mathbf{L}_{(L+i^*)} \widetilde{\mathbf{D}^{t_0^*}} \mathbf{R}_{(L+i^*)}$ or of the form $\mathbf{L}_{(L+i^*)} \widetilde{\mathbf{D}^{t_0^*,t_1^*}} \mathbf{R}_{(L+i^*)}$, where $\widetilde{\mathbf{D}^{t_0^*,t_1^*}}$ and $\widetilde{\mathbf{D}^{t_0^*}}$ are randomly sampled matrices of the appropriate form.

We structure the rest of the proof as follows. For clarity of exposition, we first state what the challenger in the real inter-column security game should generate should generate. We then state how the algorithm $\mathcal{B}$ simulates the same in a manner that is either computationally or statistically indistinguishable from the real game.

Real World: Aggregate Matrices. Let $P_1, \ldots, P_T$ be the oblivious branching programs of depth L that are to be handled in parallel, where for each $t \in [T]$, we have

$$P_t = \{M_{t,\ell,0}, M_{t,\ell,1}\}_{\ell \in [L]}.$$

The challenger in the real inter-column security game uniformly randomly samples *additional permutation matrices* $Z_1, \ldots, Z_{L-1} \in \{0,1\}^{5\times 5}$, and defines a new set of matrices $\{N_{t,\ell,b}\}_{t\in[T],\ell\in[L],b\in\{0,1\}}$, where

$$N_{t,\ell,b} = Z_{\ell-1}^{-1} M_{t,\ell,b} Z_\ell,$$

where Z_0 and Z_L are both set to be the identity permutation.

Let the corresponding set of ring-embedded permutation matrices be denoted as $\{\mathbf{N}_{t,\ell,b}\}_{t\in[T],\ell\in[L],b\in\{0,1\}}$, where each $\mathbf{N}_{t,\ell,b}$ is structured as follows:

$$\mathbf{N}_{t,\ell,b} = \begin{bmatrix} \mathbf{N}_{t,\ell,b,1,1} & \cdots & \mathbf{N}_{t,\ell,b,1,5} \\ \vdots & \ddots & \vdots \\ \mathbf{N}_{t,\ell,b,5,1} & \cdots & \mathbf{N}_{t,\ell,b,5,5} \end{bmatrix}.$$

For each program index $t \in [T]$, each level $\ell \in [L]$ and each bit $b \in \{0,1\}$, the challenger in the real inter-column security game generates an "aggregate" block-diagonal matrix $\mathbf{A}_{\ell,b,w,v}^{t_0^*,t_1^*}$, structured as:

$$\mathbf{A}_{t,\ell,b}^{t_0^*,t_1^*} = \begin{bmatrix} \mathbf{N}_{1,\ell,b,w,v} & & \\ & \ddots & \\ & & \mathbf{N}_{T,\ell,b,w,v} \end{bmatrix}.$$

Real World: Program-Carrying Matrices. Based on the above, for each level $\ell \in [L]$ and for each bit $b \in \{0,1\}$, the challenger in the real inter-column security game generates a set of $5 \cdot 5$ block diagonal 'program-carrying matrices" $\{\mathbf{P}^{t_0^*,t_1^*}_{\ell,b,w,v}\}_{w,v\in[5]}$, each of dimension $(2(L+N)+T)m \times (2(L+N)+1)m$, structured as:

$$\mathbf{P}^{t_0^*,t_1^*}_{\ell,b,w,v} = \begin{bmatrix} \mathbf{A}^{t_0^*,t_1^*}_{\ell,b,w,v} & & & & & \\ & \mathbf{C}_{(\ell,1),(b,0),(w,v)} & & & & \\ & & \mathbf{C}_{(\ell,1),(b,1),(w,v)} & & & \\ & & & \ddots & & \\ & & & & \mathbf{C}_{(\ell,L+N),(b,0),(w,v)} & \\ & & & & & \mathbf{C}_{(\ell,L+N),(b,1),(w,v)} \end{bmatrix},$$

where, for each $\ell \in [L]$, each $\ell' \in [L+N]$, each $b, b' \in \{0,1\}$ and each matrix position $(w,v) \in [5] \times [5]$, we have

$$\mathbf{C}_{(\ell,\ell'),(b,b'),(w,v)} = \begin{cases} \mathbf{0} & \text{if } (\ell',b') = (\ell, 1-b), \\ \text{uniform in } R^{m\times m} & \text{otherwise.} \end{cases}.$$

Simulation: Program-Carrying Matrices. We now focus on how $\mathcal{B}$ simulates the "guarded" versions of the program-carrying matrices using its own input elements. Note that for simulating the consistency check block submatrices, $\mathcal{B}$ simply uses the generator elements that is receives as part if its input. When generating a random "guarded" block submatrix in a given diagonal slot, it subset-sums sufficiently many instances of the corresponding generator matrices, which results in the desired distribution by the leftover hash lemma. A zero "guarded" block submatrix in a given diagonal slot, on the other hand, conceptually corresponds to subset-summing the appropriate generator elements with an "all-zero" bit string, or equivalently, not summing any of these elements.

For simulating the block submatrices corresponding to the "aggregate matrix", the natural stragey for $\mathcal{B}$ to adopt is to appropriately subset-sum the generator elements of the form $\mathbf{L}_\ell\widetilde{\mathbf{D}^t}\mathbf{R}_\ell$ that is receives as part if its input. Note that this strategy would work for all program indices other than the challenge program index t_1^*, since $\mathcal{B}$ does not receive any generator elements of the form $\mathbf{L}_\ell\widetilde{\mathbf{D}^{t_1^*}}\mathbf{R}_\ell$ for any $\ell \in [L]$.

However, in this case, $\mathcal{B}$ leverages the fact that for the challenge program indices t_0^*, t_1^* in the inter-column security game, the corresponding programs $P_{t_0^*}$ and $P_{t_1^*}$ must have *identical sets of permutation matrices* at each level, and hence identical sets of Killian-style randomized permutation matrices at each level. In other words, for $P_{t_0^*}$ and $P_{t_1^*}$, $\mathcal{B}$ can use the same ring-embedded permutation matrix representations, which allows it to use the challenge-relevant elements of the form $\mathbf{L}_\ell\widetilde{\mathbf{D}^{t_0^*,t_1^*}}\mathbf{R}_\ell$.

To summarize, $\mathcal{B}$ uses the following strategy at each level $\ell \in [L]$ for simulating the block submatrices corresponding to the "aggregate matrix":

1. For simulating the block submatrices corresponding to any program index $t \in [N] \setminus \{t_0^*, t_1^*\}$, $\mathcal{B}$ appropriately subset-sums the generator elements of the form $\mathbf{L}_\ell\widetilde{\mathbf{D}^t}\mathbf{R}_\ell$ that it receives as input.

2. For simulating the block submatrices corresponding to the program indices t_0^*, t_1^*, $\mathcal{B}$ appropriately subset-sums the challenge-relevant elements of the form $\mathbf{L}_\ell\widetilde{\mathbf{D}^{t_0^*,t_1^*}}\mathbf{R}_\ell$ that it receives as input.

It is easy to see that by the leftover hash lemma, $\mathcal{B}$'s simulation is statistically indistinguishable from the real inter-column security game.

Real World: Enforcer Matrices. For each variable index $i \in [N]$ and for each bit $b \in \{0,1\}$, the challenger in the real inter-column security game generates an additional "enforcer matrix" $\mathbf{E}^{t_0^*,t_1^*}_{i,b}$ of dimension $(2(L+N)+T)m \times$

$(2(L+N)+T)m$, structured as:

$$\mathbf{E}_{i,b}^{t_0^*,t_1^*} = \begin{bmatrix} \mathbf{T}_{i,b} & & & & & \\ & \mathbf{C}_{(i,1),(b,0)} & & & & \\ & & \mathbf{C}_{(i,1),(b,1)} & & & \\ & & & \ddots & & \\ & & & & \mathbf{C}_{(i,L+N),(b,0)} & \\ & & & & & \mathbf{C}_{(i,L+N),(b,1)} \end{bmatrix},$$

where for each $i \in [N]$, each $\ell' \in [L+N]$, and each $b, b' \in \{0,1\}$, we have

$$\mathbf{C}_{(\ell,\ell'),(b,b')} = \begin{cases} \mathbf{0} & \text{if } (i,b') = (\phi\ell, b), \\ \text{uniform in } R^{m\times m} & \text{otherwise.} \end{cases},$$

where recall that $\phi : [L] \to [N]$ is a mapping such that for each $\ell \in [L]$, $i = \phi(\ell)$ indicates which variable x_i controls the ℓ^{th} level branch.

Simulation: Enforcer Matrices. We now focus on how $\mathcal{B}$ simulates the "guarded" versions of these enforcer matrices using its own input elements. In the simulation, $\mathcal{B}$ structures the "activation-enforcer" matrix $\mathbf{T}_{i,b}$ as a block diagonal matrix in $R^{Tm\times Tm}$, as described below:

$$\mathbf{T}_{i,b} = \begin{bmatrix} \mathbf{T}_{1,i,b} & & \\ & \ddots & \\ & & \mathbf{T}_{T,i,b} \end{bmatrix},$$

where for each program index $t \in [T] \setminus \{t_0^*, t_1^*\}$, the block matrix $\mathbf{T}_{t,i,b}$ is simulated so as to satisfy the following distribution:

$$\mathbf{T}_{t,i,b} = \begin{cases} \mathbf{0} & \text{if } \mathbf{G}[i,t,b] = 0, \\ \text{uniform in } R^{m\times m} & \text{otherwise.} \end{cases}.$$

We begin by noting that by appropriately subset-summing over the generator elements received as input, $\mathcal{B}$ can simulate "guarded" versions of all consistency sub-matrices and all block submatrices in the "activation-enforcer" matrix $\mathbf{T}_{i,b}$, except for the block submatrices corresponding to the diagonal slot t_1^*. The reason $\mathcal{B}$ cannot simulate this submatrix is that by definition of the subspace hiding assumption, it is not provided with the generator elements corresponding to the diagonal slot t_1^*.

Hence, the crux of the reduction lies in how $\mathcal{B}$ simulates "guarded" versions of the block sub-matrices $\mathbf{T}_{t_0^*,i,b}$ and $\mathbf{T}_{t_1^*,i,b}$ for different values of (i,b). This simulation is broken into several sub-cases depending on the values of $(i,b) \in [N] \times \{0,1\}$, as described below:

1. **Case-1:** $i \neq i^*$ **and** $\mathbf{G}[i,t_0^*,b] = \mathbf{G}[i,t_1^*,b] = 0$:

   In this case, $\mathcal{B}$ simulates $\mathbf{T}_{t_0^*,i,b}$ and $\mathbf{T}_{t_1^*,i,b}$ as zero matrices in $R^{m\times m}$.

2. **Case-2:** $i \neq i^*$**,** $\mathbf{G}[i,t_0^*,b] = 1$**,** $\mathbf{G}[i,t_1^*,b] = 0$:

   In this case, $\mathcal{B}$ simulates $\mathbf{T}_{t_0^*,i,b}$ as a uniformly distributed matrix in $R^{m\times m}$ and $\mathbf{T}_{t_1^*,i,b}$ as the zero matrix in $R^{m\times m}$. To do this, it additionally subset-sums over generator matrices of the form $\mathbf{L}_{L+i}\widetilde{\mathbf{D}^{t_0^*}}\mathbf{R}_{L+i}$, which it is provided with as input.

3. **Case-3: $i \neq i^*$, $\mathbf{G}[i, t_0^*, b] = \mathbf{G}[i, t_1^*, b] = 1$:**

   In this case, $\mathcal{B}$ simulates both $\mathbf{T}_{t_0^*,i,b}$ and $\mathbf{T}_{t_1^*,i,b}$ as uniformly distributed matrices in $R^{m \times m}$. To do this, it additionally subset-sums over the challenge-relevant matrices of the form $\mathbf{L}_{L+i}\widetilde{\mathbf{D}^{t_0^*,t_1^*}}\mathbf{R}_{L+i}$, which it is provided with as input.

4. **Case-4: $(i, b) = (i^*, 1 - b^*)$, $\mathbf{G}[i, t_0^*, b] = \mathbf{G}[i, t_1^*, b] = 0$:**

   In this case, $\mathcal{B}$ simulates both $\mathbf{T}_{t_0^*,i,b}$ and $\mathbf{T}_{t_1^*,i,b}$ as the zero matrix in $R^{m \times m}$.

5. **Case-5: $(i, b) = (i^*, 1 - b^*)$, $\mathbf{G}[i, t_0^*, b] = 1$, $\mathbf{G}[i, t_1^*, b] = 0$:**

   In this case, $\mathcal{B}$ simulates $\mathbf{T}_{t_0^*,i,b}$ as a uniformly distributed matrix in $R^{m \times m}$ and $\mathbf{T}_{t_1^*,i,b}$ as the zero matrix in $R^{m \times m}$. To do this, it additionally subset-sums over generator matrices of the form $\mathbf{L}_{i^*+N}\widetilde{\mathbf{D}^{t_0^*}}\mathbf{R}_{i^*+N}$, which it is provided with as input.

6. **Case-6: $(i, b) = (i^*, 1 - b^*)$, $\mathbf{G}[i, t_0^*, b] = 0$, $\mathbf{G}[i, t_1^*, b] = 1$:**

   In this case, $\mathcal{B}$ simulates $\mathbf{T}_{t_0^*,i,b}$ as the zero matrix in $R^{m \times m}$ and $\mathbf{T}_{t_1^*,i,b}$ as a uniformly distributed matrix in $R^{m \times m}$. To do this, it additionally subset-sums over the special challenge-index matrices of the form $\mathbf{L}_{i^*+N}\widetilde{\mathbf{D}^{t_1^*}}\mathbf{R}_{i^*+N}$, which it is provided with as input.

7. **Case-7: $(i, b) = (i^*, 1 - b^*)$, $\mathbf{G}[i, t_0^*, b] = \mathbf{G}[i, t_1^*, b] = 1$:**

   In this case, $\mathcal{B}$ simulates both $\mathbf{T}_{t_0^*,i,b}$ and $\mathbf{T}_{t_1^*,i,b}$ as uniformly distributed matrices in $R^{m \times m}$. To do this, it additionally subset-sums over generator matrices of the form $\mathbf{L}_{i^*+N}\widetilde{\mathbf{D}^{t_0^*}}\mathbf{R}_{i^*+N}$ as well as the special challenge-index matrices of the form $\mathbf{L}_{i^*+N}\widetilde{\mathbf{D}^{t_1^*}}\mathbf{R}_{i^*+N}$, all of which it is provided with as input.

8. **Case-8: $(i, b) = (i^*, b^*)$:**

   In this case, we must have $\mathbf{G}[i^*, t_0^*, b^*] = 1$. In this case, $\mathcal{B}$ uses the challenge term provided to it as input.

Putting Everything Together. Finally, observe the following:

1. If the challenge sample is of the form $\mathbf{L}_{(L+i^*)}\widetilde{\mathbf{D}^{t_0^*}}\mathbf{R}_{(L+i^*)}$, then Case-3 would correspond to the case where $\mathbf{G}[i^*, t_1^*, b^*] = 0$.

2. If the challenge sample is of the form $\mathbf{L}_{(L+i^*)}\widetilde{\mathbf{D}^{t_0^*,t_1^*}}\mathbf{R}_{(L+i^*)}$, then Case-3 would correspond to the case where $\mathbf{G}[i^*, t_1^*, b^*] = 1$. □

Thus $\mathcal{B}$ can leverage $\mathcal{A}'s$ non-negligible advantage in the inter-column security game to achieve a non-negligible advantage in breaking the subspace hiding assumption. This completes the proof of inter-column security.

7.2 Single-Input Program Switching Security

Lemma 7.2. *Assuming that the subspace hiding assumption holds, our iaiO construction achieves single-input program switching security.*

Proof. We will prove this via a hybrid argument that incrementally "erases" the branching program matrices not corresponding to the single relevant input (referred to as $\mathbf{x}^* = (x_1^*, \ldots, x_N^*)$) in the relevant slots using the subspace hiding assumption. Once we have done this, we can argue information-theoretically to switch the programs and then reverse the hybrid to insert the new matrices.

We define the following hybrid experiments:

1. Exp_0 will denote the original program switching security game with the challenge bit set to 0 (here the original challenge program P_{t^*} is used in the $t^{*\text{th}}$ slot of each "program-carrying" matrix.

2. For each $\ell \in [L]$, we define Exp_ℓ to be identical to Exp_0, except for the first ℓ positions of each branching program, where the corresponding program-carrying submatrices corresponding to P_{t^*} for the bit values disagreeing with the single relevant input $\mathbf{x}^* = (x_1^*, \ldots, x_N^*)$ are set to the all-zero matrix in $R^{m \times m}$.

Note that in Exp_L, any pair of "program-carrying" matrices at a given level has exactly one non-zero ring-embedded permutation matrix in each slot.

We first argue that for each $\ell \in [L]$, Exp_ℓ is indistinguishable from $\mathsf{Exp}_{\ell-1}$, under the subspace hiding assumption. To see this, suppose there is some $\ell^* \in [L]$ such that some PPT attacker $\mathcal{A}$ distinguishes Exp_{ℓ^*} from Exp_{ℓ^*-1} with non-negligible advantage for some valid setting of activation matrix $\mathbf{G}$, challenge program index $t^* \in [T]$, single relevant input $\mathbf{x}^* = (x_1^*, \ldots, x_N^*)$, and challenge bit $b^* \in \{0, 1\}$. We use $\mathcal{A}$ to build a PPT algorithm $\mathcal{B}$ that breaks the subspace hiding assumption (parameterized in this case by the challenge-index $j^* = \ell^*$ and the challenge-slots $(i_0^*, i_1^*) = (t^*, T + 2\ell^* + b^*)$) with non-negligible advantage.

Inputs to $\mathcal{B}$. As described in Section 6, $\mathcal{B}$ receives as input several terms of the following kinds:

- **Generators:** For each $\ell \in [L + N]$ and each $j \in [2(L + N) + T] \setminus \{t^*\}$: many samples of the form $\mathbf{L}_\ell \widetilde{\mathbf{D}^j} \mathbf{R}_\ell$, where $\widetilde{\mathbf{D}^j}$ is a randomly sampled matrix of the appropriate form, and the "end guards" $\mathbf{L}_1$ and $\mathbf{R}_{L+N}$ are specially structured, as described in Section 5.2.

- **Challenge-Relevant Elements**: For each $\ell \in [L + N]$: many samples of the form $\mathbf{L}_\ell \mathbf{D}^{t^*, \widetilde{(T+2\ell^*+b^*)}} \mathbf{R}_\ell$, where $\mathbf{D}^{t^*, \widetilde{(T+2\ell^*+b^*)}}$ is a randomly sampled matrix of the appropriate form, and the "end guards" $\mathbf{L}_1$ and $\mathbf{R}_{L+N}$ are specially structured as mentioned before.

- **Special Challenge-Index Elements**: Many samples of the form $\mathbf{L}_{\ell^*} \widetilde{\mathbf{D}^{t^*}} \mathbf{R}_{\ell^*}$, where $\widetilde{\mathbf{D}^{t^*}}$ is a randomly sampled matrix of the appropriate form.

- **Challenge Elements**: A sample that is either of the form $\mathbf{L}_{\ell^*} \widetilde{\mathbf{D}^{t^*}} \mathbf{R}_{\ell^*}$ or of the form $\mathbf{L}_{\ell^*} \mathbf{D}^{t^*, \widetilde{(T+2\ell^*+b^*)}} \mathbf{R}_{\ell^*}$, where $\mathbf{D}^{t^*, \widetilde{(T+2\ell^*+b^*)}}$ and $\widetilde{\mathbf{D}^{t^*}}$ are randomly sampled matrices of the appropriate form.

Real World: Program-Carrying Matrices. For each level $\ell \in [L]$ and for each bit $b \in \{0, 1\}$, the challenger in the real program switching security game generates a set of $5 \cdot 5$ block diagonal 'program-carrying matrices"

$\{\mathbf{P}_{\ell,b,w,v}\}_{w,v\in[5]}$, each of dimension $(2(L+N)+T)m \times (2(L+N)+1)m$, structured as:

$$\mathbf{P}_{\ell,b,w,v} = \begin{bmatrix} \mathbf{A}_{\ell,b,w,v} & & & & & \\ & \mathbf{C}_{(\ell,1),(b,0),(w,v)} & & & & \\ & & \mathbf{C}_{(\ell,1),(b,1),(w,v)} & & & \\ & & & \ddots & & \\ & & & & \mathbf{C}_{(\ell,L+N),(b,0),(w,v)} & \\ & & & & & \mathbf{C}_{(\ell,L+N),(b,1),(w,v)} \end{bmatrix},$$

where for each $\ell \in [L]$, each $\ell' \in [L+N]$, each $b, b' \in \{0,1\}$ and each matrix position $(w,v) \in [5]\times[5]$, we have

$$\mathbf{C}_{(\ell,\ell'),(b,b'),(w,v)} = \begin{cases} \mathbf{0} & \text{if } (\ell',b') = (\ell, 1-b), \\ \text{uniform in } R^{m\times m} & \text{otherwise.} \end{cases}.$$

Simulation: Program-Carrying Matrices. In this simulation, for each program index $t \in [T]$, each level $\ell \in [L]$ and each bit $b \in \{0,1\}$, $\mathcal{B}$ simulates an "aggregate" block-diagonal matrix $\mathbf{A}_{\ell,b,w,v}$, structured as:

$$\mathbf{A}_{t,\ell,b} = \begin{bmatrix} \mathbf{N}_{1,\ell,b,w,v} & & \\ & \ddots & \\ & & \mathbf{N}_{T,\ell,b,w,v} \end{bmatrix}.$$

except for the block submatrix corresponding to the diagonal slot t^*. We explain later how the block submatrices corresponding to the diagonal slot t^* are handled.

We now describe how $\mathcal{B}$ "actually" simulates "guarded" versions of the aforementioned program-carrying matrices using its own input elements. We begin by noting that by appropriately subset-summing over the generator elements received as input, $\mathcal{B}$ can simulate "guarded" versions of all consistency sub-matrices and all block submatrices in the "aggregate" block-diagonal matrix, except for the block submatrix corresponding to the diagonal slot t^*. The reason $\mathcal{B}$ cannot simulate this submatrix is that by definition of the subspace hiding assumption, it is not provided with the generator elements corresponding to the diagonal slot t^*.

Hence, the crux of the reduction lies in how $\mathcal{B}$ simulates the block matrix $\mathbf{N}_{t^*,\ell,b,w,v}$ for different values of (ℓ,b). This simulation is broken into several sub-cases depending on the values of $(\ell,b) \in [L]\times\{0,1\}$, as described below (recall that $\phi : [L] \to [N]$ is a mapping such that for each $\ell \in [L]$, $i = \phi(\ell)$ indicates which variable x_i controls the ℓ^{th} level branch.):

1. **Case-1: $\ell \neq \ell^*$ and $b = x^*_{\phi(\ell)}$:**

   In this case, $\mathcal{B}$ should set $\mathbf{N}_{t^*,\ell,b,w,v}$ to either zero or uniformly random as per the original program and the consistency check matrix $\mathbf{C}_{\ell,1-b,(w,v)}$ to the zero matrix in $R^{m\times m}$. In particular the consistency check matrix $\mathbf{C}_{\ell^*,b^*,(w,v)}$ is uniformly random. Hence, to simulate this, $\mathcal{B}$ can appropriately subset-sum over the following input elements: (a) the challenge-relevant elements $\mathbf{L}_\ell \mathbf{D}^{t^*,\widetilde{(T+2\ell^*+b^*)}}\mathbf{R}_\ell$, and (b) all other generator elements, *including* the generator element $\mathbf{L}_\ell \mathbf{D}^{\widetilde{T+2\ell^*+b^*}}\mathbf{R}_\ell$.

2. **Case-2: $\ell > \ell^*$ and $b = 1 - x^*_{\phi(\ell)}$:**

   Note that even in this case, $\mathcal{B}$ should set $\mathbf{N}_{t^*,\ell,b,w,v}$ to either zero or uniformly random as per the original program and the consistency check matrix $\mathbf{C}_{\ell,1-b,(w,v)}$ to the zero matrix in $R^{m\times m}$. In particular the consistency check matrix $\mathbf{C}_{\ell^*,b^*,(w,v)}$ is uniformly random. Hence, to simulate this, $\mathcal{B}$ can appropriately subset-sum over the following input elements: (a) the challenge-relevant elements $\mathbf{L}_\ell \mathbf{D}^{t^*,\widetilde{(T+2\ell^*+b^*)}}\mathbf{R}_\ell$, and (b) all other generator elements, *including* the generator element $\mathbf{L}_\ell \mathbf{D}^{\widetilde{T+2\ell^*+b^*}}\mathbf{R}_\ell$.

3. **Case-3: $\ell < \ell^*$ and $b = 1 - x^*_{\phi(\ell)}$**:

   In this case, $\mathcal{B}$ should set $\mathbf{N}_{t^*,\ell,b,w,v}$ to zero (as previous hybrids have eliminated this program matrix), while the consistency check matrix $\mathbf{C}_{\ell^*,b^*,(w,v)}$ should be uniformly random. Hence, to simulate this, it suffices for $\mathcal{B}$ to appropriately subset-sum over only the generator elements it received as input, *including* the generator element $\mathbf{L}_\ell \widetilde{\mathbf{D}^{T+2\ell^*+b^*}} \mathbf{R}_\ell$..

4. **Case-4: $\ell = \ell^*$ and $b = 1 - b^*$**:

   In this case, $\mathcal{B}$ should set $\mathbf{N}_{t^*,\ell,b,w,v}$ to either zero or uniformly random as per the original program and the consistency check matrix $\mathbf{C}_{\ell^*,b^*,(w,v)}$ to the zero matrix in $R^{m\times m}$. Hence, to simulate this, $\mathcal{B}$ can appropriately subset-sum over the following input elements: (a) the "special" challenge-index elements $\mathbf{L}_{\ell^*} \widetilde{\mathbf{D}^{t^*}} \mathbf{R}_{\ell^*}$, and (b) all generator elements *excluding* the generator element $\mathbf{L}_\ell \widetilde{\mathbf{D}^{T+2\ell^*+b^*}} \mathbf{R}_\ell$.

5. **Case-5: $\ell = \ell^*$ and $b = b^*$**:

   In this case, $\mathcal{B}$ uses the challenge term provided to it as input. Note that this is an acceptable simulation strategy as $\mathcal{B}$ should set the consistency check matrix $\mathbf{C}_{\ell^*,b^*,(w,v)}$ to uniformly random in this case.

At this point, observe the following:

1. If the challenge sample is of the form $\mathbf{L}_{\ell^*} \widetilde{\mathbf{D}^{t^*}} \mathbf{R}_{\ell^*}$, then so far, $\mathcal{B}$ has properly simulated "guarded" versions of the program-carrying matrices as per Expt_{ℓ^*}.

2. If the challenge sample is of the form $\mathbf{L}_{\ell^*} \mathbf{D}^{t^*,\widetilde{(T+2\ell^*+b^*)}} \mathbf{R}_{\ell^*}$, then so far, $\mathcal{B}$ has properly simulated "guarded" versions of the program-carrying matrices as per Expt_{ℓ^*-1}.

Real World: Enforcer Matrices. For each variable index $i \in [N]$ and for each bit $b \in \{0,1\}$, the challenger in the real program switching security game generates an additional "enforcer matrix" $\mathbf{E}_{i,b}$ of dimension $(2(L+N)+T)m \times (2(L+N)+T)m$, structured as:

$$\mathbf{E}_{i,b} = \begin{bmatrix} \mathbf{T}_{i,b} & & & & & \\ & \mathbf{C}_{(i,1),(b,0)} & & & & \\ & & \mathbf{C}_{(i,1),(b,1)} & & & \\ & & & \ddots & & \\ & & & & \mathbf{C}_{(i,L+N),(b,0)} & \\ & & & & & \mathbf{C}_{(i,L+N),(b,1)} \end{bmatrix},$$

where for each $i \in [N]$, each $\ell' \in [L+N]$, and each $b, b' \in \{0,1\}$, we have

$$\mathbf{C}_{(\ell,\ell'),(b,b')} = \begin{cases} \mathbf{0} & \text{if } (i,b') = (\phi\ell, b), \\ \text{uniform in } R^{m\times m} & \text{otherwise.} \end{cases},$$

Simulation: Enforcer Matrices. In this simulation, $\mathcal{B}$ structures the "activation-enforcer" matrix $\mathbf{T}_{i,b}$ as a block diagonal matrix in $R^{Tm\times Tm}$, as described below:

$$\mathbf{T}_{i,b} = \begin{bmatrix} \mathbf{T}_{1,i,b} & & \\ & \ddots & \\ & & \mathbf{T}_{T,i,b} \end{bmatrix},$$

where for each program index $t \in [T] \setminus \{t^*\}$, the block matrix $\mathbf{T}_{t,i,b}$ is simulated as follows:

$$\mathbf{T}_{t,i,b} = \begin{cases} \mathbf{0} & \text{if } \mathbf{G}[i,t,b] = 0, \\ \text{uniform in } R^{m \times m} & \text{otherwise.} \end{cases},$$

We explain later how the block submatrices corresponding to the challenge program index t^* are handled.

We now describe how $\mathcal{B}$ "actually" simulates "guarded" versions of the aforementioned enforcer matrices using its own input elements. We begin by noting that by appropriately subset-summing over the generator elements received as input, $\mathcal{B}$ can simulate "guarded" versions of all consistency sub-matrices and all block submatrices in the "activation-enforcer" matrix $\mathbf{T}_{i,b}$, except for the block submatrix corresponding to the diagonal slot t^*. The reason $\mathcal{B}$ cannot simulate this submatrix is that by definition of the subspace hiding assumption, it is not provided with the generator elements corresponding to the diagonal slot t^*.

Hence, the crux of the reduction lies in how $\mathcal{B}$ simulates the enforcer block matrices $\mathbf{T}_{t^*,i,b}$ for different values of (i,b). This simulation is broken into several sub-cases depending on the values of $(i,b) \in [N] \times \{0,1\}$, as described below:

1. <u>**Case-1:** $\mathbf{G}[i,t^*,b] = 0$</u>:

   In this case, $\mathcal{B}$ should set $\mathbf{T}_{t^*,i,b}$ to the zero matrix in $R^{m \times m}$. Hence, to simulate this, it suffices for $\mathcal{B}$ to appropriately subset-sum over only the generator elements it received as input, *including* the generator element $\mathbf{L}_{L+i}\widetilde{\mathbf{D}^{T+2\ell^*+b^*}}\mathbf{R}_{L+i}$.

2. <u>**Case-2:** $\mathbf{G}[i,t^*,b] = 1$</u>:

   In this case, it must be that either $\phi(\ell^*) \neq i$ or $b^* \neq b$. This is because, in column t^* and row i of the activation matrix $\mathbf{G}$, there can be only one entry equal to 1 (since exactly one input is activated), and the entry $\mathbf{G}[\phi(\ell^*), t^*, 1-b^*] = 1$ by definition of b^*. Hence, in these cases, $\mathcal{B}$ should set $\mathbf{T}_{t^*,i,b}$ to a uniformly distributed matrix in $R^{m \times m}$.

   To simulate this, we claim that $\mathcal{B}$ can appropriately subset-sum over the following input elements: (a) the challenge-relevant elements $\mathbf{L}_{L+i}\mathbf{D}^{t^*,\widetilde{(T+2\ell^*+b^*)}}\mathbf{R}_{L+i}$, and (b) all generator elements *including* the generator element $\mathbf{L}_{L+i}\widetilde{\mathbf{D}^{T+2\ell^*+b^*}}\mathbf{R}_{L+i}$.

To see that $\mathcal{B}$ uses a sound simulation strategy in Case-2, observe that in this case, it is not a problem if the consistency check matrix $\mathbf{C}_{\ell^*,b^*}$ is set to a uniformly distributed matrix in $R^{m \times m}$, as either $\phi(\ell^*) \neq i$ or $b^* \neq b$. Recall that in our parallel iO construction, for any enforcer matrix $\mathbf{E}'_{i,b}$, a consistency check submatrix $\mathbf{C}_{\ell,b'}$ should be set to a uniformly distributed matrix in $R^{m \times m}$ if either $\phi(\ell) \neq i$ or $b \neq b'$.

Putting Everything Together. Finally, observe the following:

1. If the challenge sample is of the form $\mathbf{L}_{\ell^*}\widetilde{\mathbf{D}^{t^*}}\mathbf{R}_{\ell^*}$, then $\mathcal{B}$ has properly simulated "guarded" versions of the program-carrying matrices *and* the enforcer matrices as per Expt_{ℓ^*}.

2. If the challenge sample is of the form $\mathbf{L}_{\ell^*}\mathbf{D}^{t^*,\widetilde{(T+2\ell^*+b^*)}}\mathbf{R}_{\ell^*}$, then $\mathcal{B}$ has properly simulated "guarded" versions of the program-carrying matrices *and* the enforcer matrices as per Expt_{ℓ^*-1}.

This establishes the computational indistinguishability of Expt_{ℓ^*-1} Expt_{ℓ^*}. The computational indistinguishability of Expt_0 and Expt_L follows by a simple hybrid argument.

Next, we argue that the distribution of Expt_L is *statistically* close to the distribution of another experiment Expt'_L, where the challenge program P_{t^*} replaced by another program P'_{t^*} of the same depth and input access pattern that agrees with P_{t^*} on the single activated input. This follows from the following lemma:

Lemma 7.3. *Let* $\mathbf{x} = (x_1, \ldots, x_N)$ *denote a single input to a matrix branching program* $P = \{\mathbf{M}_{\ell,b}\}_{\ell\in[L],b\in\{0,1\}}$*, and suppose that we "erase" the subset of permutation matrices that do not correspond to the single input* $\mathbf{x}$*, i.e., the following set of permutation matrices is "erased":*

$$\{\mathbf{M}_{\ell,1-x_{\phi(\ell)}}\}_{\ell\in[L]}.$$

Then if $Z_1, \ldots, Z_{L-1} \in \{0,1\}^{5\times 5}$ *are uniformly sampled permutation matrices, the distribution of the following set of permutation matrices:*

$$\{N_{\ell,x_{\phi(\ell)}} = Z_{\ell-1}^{-1} M_{\ell,x_{\phi(\ell)}} Z_\ell\}_{\ell\in[L]},$$

depends only on the output of the branching program P *when evaluated on the single input* $\mathbf{x}$*.*

Given the aforementioned information-theoretic transition from Expt_L to Expt'_L, we can apply a similar sequence of hybrid steps to "reverse-transition" computationally from Expt'_L to Expt'_0, by gradually restoring the deleted permutation sub-matrices, albeit with respect to the challenge program P'_{t^*} instead of P_{t^*}.

This completes our proof of single-input program switching security.

7.3 Completely Inactive Program Security and Intra-Column Security

Lemma 7.4. *Assuming that the subspace hiding assumption holds, our iaiO construction achieves completely inactive program security.*

Proof. Leveraging techniques introduced in [GLSW15], we can show that the proof of completely inactive program security follows from the same hybrid arguments as used in the proof of single input switching security.

Note that in the completely inactive program security game, we "erase" the permutation sub-matrices corresponding to the challenge program P_{t^*} from *all* of the program-carrying matrices, and not just the matrices that do not correspond to a specific input. Once we erase all of the sub-matrices, we can again "reverse-insert" permutation sub-matrices corresponding to some other program P'_{t^*}. We can use the same hybrid arguments as in the proof of single input switching security to argue that each such transition is computationally indistinguishable

Note that unlike the proof of single input switching security, here we do not need the information-theoretic transition argument from Killian, as we are able to erase all the matrices, leaving no distribution that needs to be matched between the old program P_{t^*} and the new program P'_{t^*}.

Lemma 7.5. *Assuming that the subspace hiding assumption holds, our iaiO construction achieves intra-column security.*

Proof. Here we can again leverage the techniques presented in [GLSW15] to show that intra-column security follows from an iterative application of the arguments used in the proof of inter-column security.

8 From "Baby" Subspace Hiding Assumption To Full Assumption

In this section, we show that the baby subspace hiding assumption as in Definition 6.2 implies the general subspace hiding assumption as in definition 6.1. More concretely, we state and prove the following lemma.

Lemma 8.1. *Assuming that the baby subspace hiding assumption as in Definition 6.2 holds over a ring* R*, the general subspace hiding assumption as in definition 6.1 holds over the same ring* R*.*

Proof. To prove this lemma, we show that given a PPT algorithm $\mathcal{A}$ that breaks the general subspace hiding assumption with non-negligible advantage, one can construct a PPT algorithm $\mathcal{B}$ that breaks the baby subspace hiding assumption with non-negligible advantage. For ease of exposition, we first show the reduction for the special case where $\mathcal{A}$ chooses challenge-slots $(i_0^*, i_1^*) = (1, 2)$. We subsequently argue that the reduction works for any arbitrary choice of challenge-slots.

Inputs to $\mathcal{B}$. As per Definition 6.2, $\mathcal{B}$ receives as input several terms of the following kinds (we assume without loss of generality that $\mathcal{B}$ chooses $i_0^* = 1$ and $i_1^* = 2$):

- **Generators:** For each $j \in [z]$: many samples of the form $\mathbf{L}_j\widetilde{\mathbf{D}^1}\mathbf{R}_j$, where $\widetilde{\mathbf{D}^1}$ is a randomly sampled matrix of the appropriate form as described above, except for when $j = 1$ and $j = z$, where the samples are of the form $\mathbf{I}^2\widetilde{\mathbf{D}^1}\mathbf{R}_1$ and $\mathbf{L}_z\widetilde{\mathbf{D}^1}(\mathbf{I}^2)^T$, respectively.

- **Challenge-Relevant Elements**: For each $j \in [z]$: many samples of the form $\mathbf{L}_j\widetilde{\mathbf{D}^{1,2}}\mathbf{R}_j$, where $\widetilde{\mathbf{D}^{1,2}}$ is a randomly sampled matrix of the appropriate form, except for when $j = 1$ and $j = z$, where the samples are of the form $\mathbf{I}^2\widetilde{\mathbf{D}^{1,2}}\mathbf{R}_1$ and $\mathbf{L}_z\widetilde{\mathbf{D}^{1,2}}(\mathbf{I}^2)^T$, respectively.

- **Special Challenge-Index Elements**: Many samples of the form $\mathbf{L}_{j^*}\widetilde{\mathbf{D}^2}\mathbf{R}_{j^*}$, where $\widetilde{\mathbf{D}^2}$ is a randomly sampled matrix of the appropriate form, except for when $j^* = 1$ or $j^* = z$, where the samples are of the form $\mathbf{I}^2\widetilde{\mathbf{D}^2}\mathbf{R}_1$ or $\mathbf{L}_z\widetilde{\mathbf{D}^2}(\mathbf{I}^2)^T$, respectively.

- **The Challenge Element**: A sample that is of one of the two following forms:
  - $\mathbf{L}_{j^*}\widetilde{\mathbf{D}^1}\mathbf{R}_{j^*}$ if $b = 0$, or
  - $\mathbf{L}_{j^*}\widetilde{\mathbf{D}^{1,2}}\mathbf{R}_{j^*}$ if $b = 1$,

  where $\widetilde{\mathbf{D}^1}$ and $\widetilde{\mathbf{D}^{1,2}}$ are randomly sampled matrices of the appropriate form, except for when $j^* = 1$, where the sample is of one of the two following forms:

  - $\mathbf{I}^2\widetilde{\mathbf{D}^1}\mathbf{R}_1$ if $b = 0$, or
  - $\mathbf{I}^2\widetilde{\mathbf{D}^{1,2}}\mathbf{R}_1$ if $b = 1$,

  or when $j^* = z$, where the sample is of one of the two following forms:

  - $\mathbf{L}_z\widetilde{\mathbf{D}^1}(\mathbf{I}^2)^T$ if $b = 0$, or
  - $\mathbf{L}_z\widetilde{\mathbf{D}^{1,2}}(\mathbf{I}^2)^T$ if $b = 1$.

Simulating Inputs to $\mathcal{A}$. Observe that $\mathcal{B}$ essentially needs to simulate the terms to be provided to $\mathcal{A}$ that are not already a part of its own challenge for the baby subspace assumption. Interestingly, these terms to be simulated are, by definition, orthogonal to the terms thet $\mathcal{B}$ receives as part of its own challenge input. In other words, $\mathcal{B}$ needs to: (a) "lift" its own inputs to be part of the general subspace assumption challenge, and (b) simulate the remaining terms in the general subspace assumption challenge, which belong to orthogonal subspaces.

An Initial Attempt. Suppose the guard matrices $\{\mathbf{L}_j, \mathbf{R}_j\}$ in $\mathcal{B}$'s challenge have dimensions $m \times 2n$ and $2n \times m$, respectively. Note that $\mathcal{B}$ does not receive the actual guard matrices in the clear; they are embedded in the challenge elements. $\mathcal{B}$ generates a fresh set of guard matrices $\{\mathbf{L}_{j,\ell-2}, \mathbf{R}_{j,\ell-2}\}$ of dimensions $m \times (\ell-2)n$ and $(\ell-2)n \times m$, respectively, satisfying the same distributional and relationship requirements as in the subspace hiding assumption. The reader may refer our iaiO construction in Section 5 for the details of how this may be done efficiently. Next, suppose hypothetically, that $\mathcal{B}$ *concatenated* the guard matrices embedded in its challenge elements with the freshly simulated set of guard matrices to produce a new set of guard matrices $\{\widetilde{\mathbf{L}}_{j,\ell}, \widetilde{\mathbf{R}}_{j,\ell}\}$ as follows:

$$\widetilde{\mathbf{R}}_{j,\ell} = \begin{bmatrix} \mathbf{R}_j & \mathbf{0}_R \\ \mathbf{0}_R & \mathbf{R}_{j,\ell-2} \end{bmatrix} \quad , \quad \widetilde{\mathbf{L}}_{j,\ell} = \begin{bmatrix} \mathbf{L}_j & \mathbf{0}_R \\ \mathbf{0}_R & \mathbf{L}_{j,\ell-2} \end{bmatrix}.$$

Note that this is hypothetical since $\mathcal{B}$ is not actually provided with the set of guards $\{\mathbf{L}_j, \mathbf{R}_j\}$. Additionally, the newly constructed guard matrices are not distributed exactly as dictated by the subspace hiding assumption guards.

Nonetheless, this hypothetical set of guards are useful in the sense that they allow us to showcase how $\mathcal{B}$ can simulate the challenge to $\mathcal{A}$ by exploiting the orthogonality of various terms it receives as input and the fresh terms it needs to simulate, albeit with some departures from the desired distribution. This is described subsequently.

1. **Simulating Generators**:

   For each $j \in [z]$ and each $i \in [\ell] \setminus \{2\}$, $\mathcal{B}$ needs to simulate many samples of the form $\widetilde{\mathbf{L}}_{j,\ell}\widetilde{\mathbf{D}_\ell^i}\widetilde{\mathbf{R}}_{j,\ell}$, where $\widetilde{\mathbf{D}_\ell^i}$ is a randomly sampled matrix of the appropriate form as described above (the subscript ℓ is used to indicate that it is contains ℓ diagonal blocks), except for when $j = 1$ and $j = z$, where the samples are of the form $\mathbf{I}^\ell\widetilde{\mathbf{D}_\ell^i}\widetilde{\mathbf{R}}_{1,\ell}$ and $\widetilde{\mathbf{L}}_{z,\ell}\widetilde{\mathbf{D}_\ell^i}(\mathbf{I}^\ell)^T$, respectively. We divide the simulation strategy into the following cases:

   (a) **Case-1:** $i = 1$:

   Note that for each $j \in [z]$, we have

   $$\widetilde{\mathbf{L}}_{j,\ell}\widetilde{\mathbf{D}_\ell^1}\widetilde{\mathbf{R}}_{j,\ell} = \begin{bmatrix} \mathbf{L}_j\widetilde{\mathbf{D}^1}\mathbf{R}_j & \mathbf{0}_R \\ \mathbf{0}_R & \mathbf{0}_R \end{bmatrix},$$

   except for when $j = 1$ and $j = z$, where we have

   $$\mathbf{I}^\ell\widetilde{\mathbf{D}_\ell^1}\widetilde{\mathbf{R}}_{1,\ell} = \begin{bmatrix} \mathbf{I}^2\widetilde{\mathbf{D}^1}\mathbf{R}_1 & \mathbf{0}_R \end{bmatrix} \quad , \quad \widetilde{\mathbf{L}}_{z,\ell}\widetilde{\mathbf{D}_\ell^1}(\mathbf{I}^\ell)^T = \begin{bmatrix} \mathbf{L}_z\widetilde{\mathbf{D}^1}(\mathbf{I}^2)^T \\ \mathbf{0}_R \end{bmatrix}.$$

   So in this case, $\mathcal{B}$ can directly use the generators it receives as input for the simulation.

   (b) **Case-2:** $i \neq 1$ **and** $i \neq 2$:

   Note that for each $j \in [z]$, we have

   $$\widetilde{\mathbf{L}}_{j,\ell}\widetilde{\mathbf{D}_\ell^i}\widetilde{\mathbf{R}}_{j,\ell} = \begin{bmatrix} \mathbf{0}_R & \mathbf{0}_R \\ \mathbf{0}_R & \mathbf{L}_{j,\ell-2}\widetilde{\mathbf{D}_{\ell-2}^i}\mathbf{R}_{j,\ell-2} \end{bmatrix},$$

   except for when $j = 1$ and $j = z$, where we have

   $$\mathbf{I}^\ell\widetilde{\mathbf{D}_\ell^i}\widetilde{\mathbf{R}}_{1,\ell} = \begin{bmatrix} \mathbf{0}_R & \mathbf{I}^{\ell-2}\widetilde{\mathbf{D}^i}\mathbf{R}_{1,\ell-2} \end{bmatrix} \quad , \quad \widetilde{\mathbf{L}}_{z,\ell}\widetilde{\mathbf{D}_\ell^i}(\mathbf{I}^\ell)^T = \begin{bmatrix} \mathbf{0}_R \\ \mathbf{L}_{z,\ell-2}\widetilde{\mathbf{D}^i}(\mathbf{I}^{\ell-2})^T \end{bmatrix}.$$

   So in this case, $\mathcal{B}$ can sample the generators directly without using its own inputs.

2. **Simulating Challenge-Relevant Elements.**

   For each $j \in [z]$, $\mathcal{B}$ needs to simulate many samples of the form $\widetilde{\mathbf{L}}_{j,\ell}\widetilde{\mathbf{D}_\ell^{1,2}}\widetilde{\mathbf{R}}_{j,\ell}$, where $\widetilde{\mathbf{D}_\ell^{1,2}}$ is a randomly sampled matrix of the appropriate form as described above (the subscript ℓ is used to indicate that it is contains ℓ diagonal blocks), except for when $j = 1$ and $j = z$, where the samples are of the form $\mathbf{I}^\ell\widetilde{\mathbf{D}_\ell^{1,2}}\widetilde{\mathbf{R}}_{1,\ell}$ and $\widetilde{\mathbf{L}}_{z,\ell}\widetilde{\mathbf{D}_\ell^{1,2}}(\mathbf{I}^\ell)^T$, respectively.

   Again, note that for each $j \in [z]$, we have

   $$\widetilde{\mathbf{L}}_{j,\ell}\widetilde{\mathbf{D}_\ell^{1,2}}\widetilde{\mathbf{R}}_{j,\ell} = \begin{bmatrix} \mathbf{L}_j\widetilde{\mathbf{D}^{1,2}}\mathbf{R}_j & \mathbf{0}_R \\ \mathbf{0}_R & \mathbf{0}_R \end{bmatrix},$$

except for when $j = 1$ and $j = z$, where we have

$$\mathbf{I}^{\ell}\widetilde{\mathbf{D}_{\ell}^{1,2}}\widetilde{\mathbf{R}}_{1,\ell} = \begin{bmatrix}\mathbf{I}^2\widetilde{\mathbf{D}^{1,2}}\mathbf{R}_1 & \mathbf{0}_R\end{bmatrix} \quad , \quad \widetilde{\mathbf{L}}_{z,\ell}\widetilde{\mathbf{D}_{\ell}^{1,2}}(\mathbf{I}^{\ell})^T = \begin{bmatrix}\mathbf{L}_z\widetilde{\mathbf{D}^{1,2}}(\mathbf{I}^2)^T \\ \mathbf{0}_R\end{bmatrix}.$$

So in this case, $\mathcal{B}$ can simulate using the challenge-relevant elements it receives as input.

3. **Simulating Special Challenge-Index Elements**:

   $\mathcal{B}$ needs to simulate many samples of the form $\widetilde{\mathbf{L}}_{j^*,\ell}\widetilde{\mathbf{D}_{\ell}^2}\widetilde{\mathbf{R}}_{j^*,\ell}$, where $\widetilde{\mathbf{D}_{\ell}^i}$ is a randomly sampled matrix of the appropriate form as described above (the subscript ℓ is used to indicate that it is contains ℓ diagonal blocks), except for when $j^* = 1$ or $j^* = z$, where the samples are of the form $\mathbf{I}^{\ell}\widetilde{\mathbf{D}_{\ell}^2}\widetilde{\mathbf{R}}_{1,\ell}$ or $\widetilde{\mathbf{L}}_{z,\ell}\widetilde{\mathbf{D}_{\ell}^2}(\mathbf{I}^{\ell})^T$, respectively.

   Again, note that we have

$$\widetilde{\mathbf{L}}_{j^*,\ell}\widetilde{\mathbf{D}_{\ell}^2}\widetilde{\mathbf{R}}_{j^*,\ell} = \begin{bmatrix}\mathbf{L}_{j^*}\widetilde{\mathbf{D}^2}\mathbf{R}_{j^*} & \mathbf{0}_R \\ \mathbf{0}_R & \mathbf{0}_R\end{bmatrix},$$

   except for when $j^* = 1$ or $j^* = z$, where we have

$$\mathbf{I}^{\ell}\widetilde{\mathbf{D}_{\ell}^2}\widetilde{\mathbf{R}}_{1,\ell} = \begin{bmatrix}\mathbf{I}^2\widetilde{\mathbf{D}^2}\mathbf{R}_1 & \mathbf{0}_R\end{bmatrix} \quad , \quad \widetilde{\mathbf{L}}_{z,\ell}\widetilde{\mathbf{D}_{\ell}^2}(\mathbf{I}^{\ell})^T = \begin{bmatrix}\mathbf{L}_z\widetilde{\mathbf{D}^2}(\mathbf{I}^2)^T \\ \mathbf{0}_R\end{bmatrix}.$$

   So even in this case, $\mathcal{B}$ can simulate using the special challenge-index elements it receives as input.

4. **Simulating the Challenge Element**:

   Note that we have

$$\widetilde{\mathbf{L}}_{j^*,\ell}\widetilde{\mathbf{D}_{\ell}^1}\widetilde{\mathbf{R}}_{j^*,\ell} = \begin{bmatrix}\mathbf{L}_{j^*}\widetilde{\mathbf{D}^1}\mathbf{R}_{j^*} & \mathbf{0}_R \\ \mathbf{0}_R & \mathbf{0}_R\end{bmatrix} \quad , \quad \widetilde{\mathbf{L}}_{j^*,\ell}\widetilde{\mathbf{D}_{\ell}^{1,2}}\widetilde{\mathbf{R}}_{j^*,\ell} = \begin{bmatrix}\mathbf{L}_{j^*}\widetilde{\mathbf{D}^{1,2}}\mathbf{R}_{j^*} & \mathbf{0}_R \\ \mathbf{0}_R & \mathbf{0}_R\end{bmatrix},$$

   except for when $j^* = 1$, where we have

$$\mathbf{I}^{\ell}\widetilde{\mathbf{D}_{\ell}^1}\widetilde{\mathbf{R}}_{1,\ell} = \begin{bmatrix}\mathbf{I}^2\widetilde{\mathbf{D}^1}\mathbf{R}_1 & \mathbf{0}_R\end{bmatrix} \quad , \quad \mathbf{I}^{\ell}\widetilde{\mathbf{D}_{\ell}^{1,2}}\widetilde{\mathbf{R}}_{1,\ell} = \begin{bmatrix}\mathbf{I}^2\widetilde{\mathbf{D}^{1,2}}\mathbf{R}_1 & \mathbf{0}_R\end{bmatrix},$$

   or when $j^* = z$, where we have

$$\widetilde{\mathbf{L}}_{z,\ell}\widetilde{\mathbf{D}_{\ell}^1}(\mathbf{I}^{\ell})^T = \begin{bmatrix}\mathbf{L}_z\widetilde{\mathbf{D}^1}(\mathbf{I}^2)^T \\ \mathbf{0}_R\end{bmatrix} \quad , \quad \widetilde{\mathbf{L}}_{z,\ell}\widetilde{\mathbf{D}_{\ell}^{1,2}}(\mathbf{I}^{\ell})^T = \begin{bmatrix}\mathbf{L}_z\widetilde{\mathbf{D}^{1,2}}(\mathbf{I}^2)^T \\ \mathbf{0}_R\end{bmatrix},$$

   Hence, $\mathcal{B}$ can forward its own challenge element to $\mathcal{A}$.

Re-randomizing the Guard Matrices. It remains to show how $\mathcal{B}$ can re-randomize the hypothetically created guards to have the correct distribution as per the subspace hiding assumption, while retaining the ability to simulate as illustrated above.

To do this, $\mathcal{B}$ uniformly samples an additional set of guard matrices $\{\widehat{\mathbf{L}_j}, \widehat{\mathbf{R}_j}\}$ of dimensions $m' \times 2m$ and $2m \times m'$, respectively, for some $m' > 6m \log |R|$ subject to the constraint that $\hat{\mathbf{R}}_j \hat{\mathbf{L}}_{j+1} = \mathbf{I}$ (this is done via the exact same procedure as used to sample the guard matrices in our iaiO construction). Next, $\mathcal{B}$ (hypothetically) creates a set of "mega guards" as:

$$\mathbf{R}^*_{j,\ell} = \widetilde{\mathbf{R}_{j,\ell}} \widehat{\mathbf{R}_j} \quad , \quad \mathbf{L}^*_{j,\ell} = \widehat{\mathbf{L}_j} \widetilde{\mathbf{L}_{j,\ell}}.$$

First, note that each product of the form $\mathbf{R}^*_{j,\ell} \mathbf{L}^*_{j+1,\ell}$ is distributed exactly the same as $\widetilde{\mathbf{R}_{j,\ell}} \widetilde{\mathbf{L}_{j+1,\ell}}$. So, if $\mathcal{B}$ takes all of the aforementioned simulated terms with the incorrectly distributed guards and left and right multiplies them with (where appropriate) with the $\widehat{\mathbf{L}_j}$ and $\widehat{\mathbf{R}_j}$ terms, respectively, then the overall relations remain the same.

It remains to show that the "mega guards" are distributed appropriately. Note that, by Lemma 3.7, we can write

$$\mathbf{R}^*_{j,\ell} = \begin{bmatrix} \mathbf{A}_j & \mathbf{A}_j \mathbf{X}_j + \mathbf{C}_j \end{bmatrix} \quad , \quad \mathbf{L}^*_{j,\ell} = \begin{bmatrix} -\mathbf{X}_j \mathbf{B}_j \\ \mathbf{B}_j \end{bmatrix}$$

for uniformly random $\mathbf{A}_j$, $\mathbf{X}_j$, $\mathbf{B}_j$ and $\mathbf{C}_j$, subject to the restriction that $\mathbf{C}_j \mathbf{B}_j = \mathbf{I}$. By Corollary 10.3 it follows that $\mathbf{R}^*_{j,\ell}$ and $\mathbf{R}^*_{j,\ell}$ are distributed indistinguishably from uniformly sampled random matrices with the desired product distribution.

Handling Arbitrary Challenge-Slots. Finally, we consider the general case where the adversary $\mathcal{A}$ in the subspace hiding experiment makes an arbitrary choice of challenge-slots $(i^*_0, i^*_1) \in [\ell] \times [\ell]$, as opposed to the specific choice $(i^*_0, i^*_1) = (1, 2)$. Note that the only difference is that in the general case, $\mathcal{B}$ needs to use its own challenge elements in the slots (i^*_0, i^*_1) and not in the slots $(1, 2)$. We show how this can be handled by $\mathcal{B}$.

Suppose again the guard matrices $\{\mathbf{L}_j, \mathbf{R}_j\}$ in $\mathcal{B}$'s challenge have dimensions $m \times 2n$ and $2n \times m$, respectively, and suppose these are of the form

$$\mathbf{R}_j = \begin{bmatrix} -- & \mathbf{R}_{j,1} & -- \\ -- & \mathbf{R}_{j,2} & -- \end{bmatrix} \quad , \quad \mathbf{L}_j = \begin{bmatrix} | & | \\ \mathbf{L}_{j,1} & \mathbf{L}_{j,2} \\ | & | \end{bmatrix},$$

where the submatrices have dimensions $m \times n$ and $n \times m$, respectively.

Note that $\mathcal{B}$ does not receive the actual guard matrices in the clear; they are embedded in the challenge elements. As a first step, $\mathcal{B}$ changes its strategy of generating the new set of (improperly distributed) guard matrices $\{\widetilde{\mathbf{L}}_{j,\ell}, \widetilde{\mathbf{R}}_{j,\ell}\}$. In particular, $\mathcal{B}$ generates three fresh sets of guard matrices (we assume without loss of generality that $i^*_1 > i^*_0$):

- $\{\mathbf{L}_{j,[1,i^*_0]}, \mathbf{R}_{j,[1,i^*_0]}\}$ that have dimensions $m \times (i^*_0 - 1)n$ and $(i^*_0 - 1)n \times m$, respectively,
- $\{\mathbf{L}_{j,[i^*_0+1,i^*_1-1]}, \mathbf{R}_{j,[i^*_0+1,i^*_1-1]}\}$ that have dimensions $m \times (i^*_1 - i^*_0 - 1)n$ and $(i^*_1 - i^*_0 - 1)n \times m$, respectively,
- $\{\mathbf{L}_{j,[i^*_1+1,\ell]}, \mathbf{R}_{j,[i^*_1+1,\ell]}\}$ with dimensions $m \times (\ell - i^*_1 - 1)n$ and $(\ell - i^*_1 - 1)n \times m$, respectively,

that all satisfy the same distributional and relationship requirements as in the subspace hiding assumption. Next, hypothetically, $\mathcal{B}$ *concatenates* the guard matrices embedded in its challenge elements with the freshly simulated set of guard matrices to produce a new set of block-diagonal guard matrices $\{\widetilde{\mathbf{L}}_{j,\ell}, \widetilde{\mathbf{R}}_{j,\ell}\}$ as follows:

$$\widetilde{\mathbf{R}}_{j,\ell} = \begin{bmatrix} \mathbf{R}_{j,[1,i^*_0]} & & & & \\ & \mathbf{R}_{j,1} & & & \\ & & \mathbf{R}_{j,[i^*_0+1,i^*_1-1]} & & \\ & & & \mathbf{R}_{j,2} & \\ & & & & \mathbf{R}_{j,[i^*_1+1,\ell]} \end{bmatrix},$$

and

$$\widetilde{\mathbf{L}}_{j,\ell} = \begin{bmatrix} \mathbf{L}_{j,[1,i_0^*]} & & & & \\ & \mathbf{L}_{j,1} & & & \\ & & \mathbf{L}_{j,[i_0^*+1,i_1^*-1]} & & \\ & & & \mathbf{L}_{j,2} & \\ & & & & \mathbf{L}_{j,[i_1^*+1,\ell]} \end{bmatrix}.$$

□

Note that this is again hypothetical since $\mathcal{B}$ is not actually provided with the set of guards $\{(\mathbf{L}_{j,1}, \mathbf{R}_{j,1}), (\mathbf{L}_{j,2}, \mathbf{R}_{j,2})\}$. Additionally, the newly constructed guard matrices are not distributed exactly as dictated by the subspace hiding assumption guards. Nonetheless, it serves to showcase how $\mathcal{B}$ can hypothetically "place" the guards in its own challenge from the baby subspace hiding experiment into the challenge-slots chosen by the adversary $\mathcal{A}$ in the subspace hiding experiment.

Next, it follows from arguments very similar to those presented for the specific case that this hypothetical set of guards can be used by $\mathcal{B}$ to generate the challenge to $\mathcal{A}$ by exploiting the orthogonality of various terms it receives as input and the fresh terms it needs to simulate, albeit with certain departures from the desired distribution. Finally, to repair these departures, $\mathcal{B}$ can use the same trick as before to (hypothetically) re-randomize the guards, and thus make sure that all the terms in the challenge to $\mathcal{A}$ are distributed as in the real subspace hiding experiment.

This completes the proof of Lemma 8.1.

Looking ahead to the next section, we prove in Lemma 9.1 that the baby subspace hiding assumption holds over the output ring R of a ring homomorphic synthesizer. Putting together Lemma 9.1 and Lemma 8.1, we immediately get the following lemma.

Lemma 8.2. *The subspace hiding assumption as in Definition 6.1 holds over the output ring R of a ring homomorphic synthesizer.*

This completes the proof of the subspace hiding assumption.

9 Proof of Baby Subspace Hiding Assumption

In this section, we prove that the baby subspace hiding assumption as in Definition 6.2 holds over any ring R that is the output ring of a ring-embedded homomorphic synthesizer.

9.1 Overview of Proof Strategy

Before presenting the formal proof, we provide some additional insight into the baby subspace hiding assumption and the proof strategy for the same using an example. Consider an instance of the baby subspace assumption experiment with $\ell = 2$ and $z = 4$, and suppose that the adversary $\mathcal{A}$ chooses the challenge-index $j^* = 3$. Then the challenger needs to provide $\mathcal{A}$ with terms of the following form (we use $\mathbf{X}$ generically to denote random submatrices of the appropriate dimensions:):

1. **Generators**:

   The challenger provides the adversary $\mathcal{A}$ with several generators of the following forms:

$$\mathbf{G}_1 = \begin{bmatrix} \mathbf{X} & \mathbf{0}_R \end{bmatrix} \cdot \begin{bmatrix} -- & \mathbf{R}_{1,1} & -- \\ -- & \mathbf{R}_{1,2} & -- \end{bmatrix} \quad , \quad \mathbf{G}_2 = \begin{bmatrix} | & | \\ \mathbf{L}_{2,1} & \mathbf{L}_{2,2} \\ | & | \end{bmatrix} \cdot \begin{bmatrix} \mathbf{X} & \mathbf{0}_R \\ \mathbf{0}_R & \mathbf{0}_R \end{bmatrix} \cdot \begin{bmatrix} -- & \mathbf{R}_{2,1} & -- \\ -- & \mathbf{R}_{2,2} & -- \end{bmatrix},$$

$$\mathbf{G}_3 = \begin{bmatrix} | & | \\ \mathbf{L}_{3,1} & \mathbf{L}_{3,2} \\ | & | \end{bmatrix} \cdot \begin{bmatrix} \mathbf{X} & \mathbf{0}_R \\ \mathbf{0}_R & \mathbf{0}_R \end{bmatrix} \cdot \begin{bmatrix} -- & \mathbf{R}_{3,1} & -- \\ -- & \mathbf{R}_{3,2} & -- \end{bmatrix} \quad , \quad \mathbf{G}_4 = \begin{bmatrix} | & | \\ \mathbf{L}_{4,1} & \mathbf{L}_{4,2} \\ | & | \end{bmatrix} \cdot \begin{bmatrix} \mathbf{X} \\ \mathbf{0}_R \end{bmatrix}$$

2. **Challenge-Relevant Elements.**

   The challenger provides the adversary $\mathcal{A}$ with several challenge-relevant elements of the following form (note that in the baby subspace assumption experiment, the challenge-slot pair is $(1,2)$ by default):

$$\mathbf{H}_1 = \begin{bmatrix} \mathbf{X} & \mathbf{X} \end{bmatrix} \cdot \begin{bmatrix} -- & \mathbf{R}_{1,1} & -- \\ -- & \mathbf{R}_{1,2} & -- \end{bmatrix} \quad , \quad \mathbf{H}_2 = \begin{bmatrix} | & | \\ \mathbf{L}_{2,1} & \mathbf{L}_{2,2} \\ | & | \end{bmatrix} \cdot \begin{bmatrix} \mathbf{X} & \mathbf{0}_R \\ \mathbf{0}_R & \mathbf{X} \end{bmatrix} \cdot \begin{bmatrix} -- & \mathbf{R}_{2,1} & -- \\ -- & \mathbf{R}_{2,2} & -- \end{bmatrix},$$

$$\mathbf{H}_3 = \begin{bmatrix} | & | \\ \mathbf{L}_{3,1} & \mathbf{L}_{3,2} \\ | & | \end{bmatrix} \cdot \begin{bmatrix} \mathbf{X} & \mathbf{0}_R \\ \mathbf{0}_R & \mathbf{X} \end{bmatrix} \cdot \begin{bmatrix} -- & \mathbf{R}_{3,1} & -- \\ -- & \mathbf{R}_{3,2} & -- \end{bmatrix} \quad , \quad \mathbf{H}_4 = \begin{bmatrix} | & | \\ \mathbf{L}_{4,1} & \mathbf{L}_{3,2} \\ | & | \end{bmatrix} \cdot \begin{bmatrix} \mathbf{X} \\ \mathbf{X} \end{bmatrix}$$

3. **Special Challenge-Index Elements.**

   The challenger provides the adversary $\mathcal{A}$ with several special challenge-index elements of the following form (note that for our example, the challenge-index $j^* = 3$):

$$\mathbf{S} = \begin{bmatrix} | & | \\ \mathbf{L}_{3,1} & \mathbf{L}_{3,2} \\ | & | \end{bmatrix} \cdot \begin{bmatrix} \mathbf{0}_R & \mathbf{0}_R \\ \mathbf{0}_R & \mathbf{X} \end{bmatrix} \cdot \begin{bmatrix} -- & \mathbf{R}_{3,1} & -- \\ -- & \mathbf{R}_{3,2} & -- \end{bmatrix}$$

4. **Challenge Element.**

   Finally, the challenger provides the adversary $\mathcal{A}$ with a challenge element of the following form (depending on the challenge-bit b):

$$\mathbf{C}_0 = \begin{bmatrix} | & | \\ \mathbf{L}_{3,1} & \mathbf{L}_{3,2} \\ | & | \end{bmatrix} \cdot \begin{bmatrix} \mathbf{X} & \mathbf{0}_R \\ \mathbf{0}_R & \mathbf{0}_R \end{bmatrix} \cdot \begin{bmatrix} -- & \mathbf{R}_{3,1} & -- \\ -- & \mathbf{R}_{3,2} & -- \end{bmatrix}$$

   or

$$\mathbf{C}_1 = \begin{bmatrix} | & | \\ \mathbf{L}_{3,1} & \mathbf{L}_{3,2} \\ | & | \end{bmatrix} \cdot \begin{bmatrix} \mathbf{X} & \mathbf{0}_R \\ \mathbf{0}_R & \mathbf{X} \end{bmatrix} \cdot \begin{bmatrix} -- & \mathbf{R}_{3,1} & -- \\ -- & \mathbf{R}_{3,2} & -- \end{bmatrix}$$

Intuitively, this assumption is secure because all elements *not* corresponding to the challenge-index are either random in the first and second slots or are random in the first slot and zero in the second slot. So all possible "legitimate" multiplications that can be performed efficiently will never zero the first slot of the challenge element, allowing it to mask the value of the second slot (where the challenge elements differ). Note that if there were an element that was random in the second slot and zero in the first, the adversary $\mathcal{A}$ could use it to trivially break security–but these elements ar not provided. We illustrate this intuition some more using the aforementioned example.

Distribution Requirements. Intuitively, we would like the terms provided by the challenger to the adversary $\mathcal{A}$ in the baby subspace assumption experiment to be computationally indistinguishable from random, while being subject to certain constraints, namely, some of them should multiply to zero by definition.

Suppose we (informally) label the elements provided by the challenger to the adversary $\mathcal{A}$ in the baby subspace assumption experiment by type and index. We label the generators as $\mathbf{G}_1, \ldots, \mathbf{G}_z$, the challenge-relevant elements as $\mathbf{H}_1, \ldots, \mathbf{H}_z$, the special challenge-index element as $\mathbf{S}$ and the final challenge element as $\mathbf{C}$. Also, let the challenge-index chosen by the adversary $\mathcal{A}$ be j^*. Note that the following relations must hold in our assumption for any $j \in [j^*]$ and any $j' \in [j^*+1, z]$, respectively:

$$\mathbf{G}_j \left(\prod_{i=j+1}^{j^*-1} \mathbf{H}_i \right) \mathbf{S} = \mathbf{0}_R \quad , \quad \mathbf{S} \left(\prod_{i=j^*+1}^{j'-1} \mathbf{H}_i \right) \mathbf{G}_{j'} = \mathbf{0}_R.$$

The astute reader may observe at this point that these are the only relations that the adversary $\mathcal{A}$ can efficiently verify. Thus, intuitively, we wish to establish that the terms provided by the challenger to the adversary $\mathcal{A}$ in the baby subspace assumption experiment are computationally indistinguishable from a set of terms that are sampled uniformly from distributions that additionally satisfy the aforementioned constraints.

Mapping to Guard Matrices. We now map the aforementioned idea onto the guard matrices embedded inside the terms provided by the challenger to the adversary $\mathcal{A}$ in the baby subspace assumption experiment. We revert back to our previous example where we considered an instance of the baby subspace assumption experiment with $\ell = 2$ and $z = 4$, with the additional assumption that the adversary $\mathcal{A}$ chooses the challenge-index $j^* = 3$. We again present the terms provided by the challenger to the adversary $\mathcal{A}$; but we additionally depict the following: we label the guard submatrices that *do not* need to satisfy constraints (i.e., submatrices which are either zeroed out or for which additional orthogonal terms are *not* provided to the adversary $\mathcal{A}$) with the color blue. In other words, the terms look as follows:

1. **Generators**:

   The challenger provides the adversary $\mathcal{A}$ with several generators of the following forms:

$$\mathbf{G}_1 = \begin{bmatrix} \mathbf{X} & \mathbf{0}_R \end{bmatrix} \cdot \begin{bmatrix} -- & \mathbf{R}_{1,1} & -- \\ -- & {\color{blue}\mathbf{R}_{1,2}} & -- \end{bmatrix} \quad , \quad \mathbf{G}_2 = \begin{bmatrix} \vert & \vert \\ {\color{blue}\mathbf{L}_{2,1}} & \mathbf{L}_{2,2} \\ \vert & \vert \end{bmatrix} \cdot \begin{bmatrix} \mathbf{X} & \mathbf{0}_R \\ \mathbf{0}_R & \mathbf{0}_R \end{bmatrix} \cdot \begin{bmatrix} -- & \mathbf{R}_{2,1} & -- \\ -- & {\color{blue}\mathbf{R}_{2,2}} & -- \end{bmatrix},$$

$$\mathbf{G}_3 = \begin{bmatrix} \vert & \vert \\ {\color{blue}\mathbf{L}_{3,1}} & \mathbf{L}_{3,2} \\ \vert & \vert \end{bmatrix} \cdot \begin{bmatrix} \mathbf{X} & \mathbf{0}_R \\ \mathbf{0}_R & \mathbf{0}_R \end{bmatrix} \cdot \begin{bmatrix} -- & {\color{blue}\mathbf{R}_{3,1}} & -- \\ -- & \mathbf{R}_{3,2} & -- \end{bmatrix} \quad , \quad \mathbf{G}_4 = \begin{bmatrix} \vert & \vert \\ \mathbf{L}_{4,1} & {\color{blue}\mathbf{L}_{4,2}} \\ \vert & \vert \end{bmatrix} \cdot \begin{bmatrix} \mathbf{X} \\ \mathbf{0}_R \end{bmatrix}$$

2. **Challenge-Relevant Elements.**

   The challenger provides the adversary $\mathcal{A}$ with several challenge-relevant elements of the following form (note that in the baby subspace assumption experiment, the challenge-slot pair is $(1, 2)$ by default):

$$\mathbf{H}_1 = \begin{bmatrix} \mathbf{X} & \mathbf{X} \end{bmatrix} \cdot \begin{bmatrix} -- & \mathbf{R}_{1,1} & -- \\ -- & {\color{blue}\mathbf{R}_{1,2}} & -- \end{bmatrix} \quad , \quad \mathbf{H}_2 = \begin{bmatrix} \vert & \vert \\ {\color{blue}\mathbf{L}_{2,1}} & \mathbf{L}_{2,2} \\ \vert & \vert \end{bmatrix} \cdot \begin{bmatrix} \mathbf{X} & \mathbf{0}_R \\ \mathbf{0}_R & \mathbf{X} \end{bmatrix} \cdot \begin{bmatrix} -- & \mathbf{R}_{2,1} & -- \\ -- & {\color{blue}\mathbf{R}_{2,2}} & -- \end{bmatrix},$$

$$\mathbf{H}_3 = \begin{bmatrix} \vert & \vert \\ {\color{blue}\mathbf{L}_{3,1}} & \mathbf{L}_{3,2} \\ \vert & \vert \end{bmatrix} \cdot \begin{bmatrix} \mathbf{X} & \mathbf{0}_R \\ \mathbf{0}_R & \mathbf{X} \end{bmatrix} \cdot \begin{bmatrix} -- & {\color{blue}\mathbf{R}_{3,1}} & -- \\ -- & \mathbf{R}_{3,2} & -- \end{bmatrix} \quad , \quad \mathbf{H}_4 = \begin{bmatrix} \vert & \vert \\ \mathbf{L}_{4,1} & {\color{blue}\mathbf{L}_{4,2}} \\ \vert & \vert \end{bmatrix} \cdot \begin{bmatrix} \mathbf{X} \\ \mathbf{X} \end{bmatrix}$$

3. **Special Challenge-Index Elements.**

   The challenger provides the adversary $\mathcal{A}$ with several special challenge-index elements of the following form (note that for our example, the challenge-index $j^* = 3$):

$$\mathbf{S} = \begin{bmatrix} | & | \\ \mathbf{L}_{3,1} & \mathbf{L}_{3,2} \\ | & | \end{bmatrix} \cdot \begin{bmatrix} \mathbf{0}_R & \mathbf{0}_R \\ \mathbf{0}_R & \mathbf{X} \end{bmatrix} \cdot \begin{bmatrix} -- & \mathbf{R}_{3,1} & -- \\ -- & \mathbf{R}_{3,2} & -- \end{bmatrix}$$

4. **Challenge Element.**

   Finally, the challenger provides the adversary $\mathcal{A}$ with a challenge element of the following form (depending on the challenge-bit b):

$$\mathbf{C}_0 = \begin{bmatrix} | & | \\ \mathbf{L}_{3,1} & \mathbf{L}_{3,2} \\ | & | \end{bmatrix} \cdot \begin{bmatrix} \mathbf{X} & \mathbf{0}_R \\ \mathbf{0}_R & \mathbf{0}_R \end{bmatrix} \cdot \begin{bmatrix} -- & \mathbf{R}_{3,1} & -- \\ -- & \mathbf{R}_{3,2} & -- \end{bmatrix}$$

or

$$\mathbf{C}_1 = \begin{bmatrix} | & | \\ \mathbf{L}_{3,1} & \mathbf{L}_{3,2} \\ | & | \end{bmatrix} \cdot \begin{bmatrix} \mathbf{X} & \mathbf{0}_R \\ \mathbf{0}_R & \mathbf{X} \end{bmatrix} \cdot \begin{bmatrix} -- & \mathbf{R}_{3,1} & -- \\ -- & \mathbf{R}_{3,2} & -- \end{bmatrix}$$

The important thing to note from the aforementioned illustration is the following: suppose that the challenger, in a sequence of hybrid experiments, replaces the "blue" guard submatrices in the aformentioned depiction with uniformly random submatrices of the same dimension. This does not violate any of the aforementioned constraints as these are the submatrices for which additional orthogonal terms are *not* provided to the adversary $\mathcal{A}$. Additionally, once all the "blue" guard submatrices have been replaced with with uniformly random submatrices of the same dimension, indistinguishability of the challenge element from a random matrix (independent of the challenge bit b) follows immediately.

Proof Strategy. Based on this observation, our proof strategy for the baby subspace hiding assumption will be to replace the "blue" guard submatrices in the aformentioned depiction with uniformly random submatrices of the same dimension, via a sequence of hybrid experiments. The proof that each hybrid is indistinguishable from the previous hybrid relies intuitively on the following observation: by invoking Theorem 4.2, we can prove that the following are computationally indistinguishable whenever the ring R is the output ring of a ring homomorphic synthesizer:

$$\begin{bmatrix} & | & \\ -- & \mathbf{X}_0 & -- \\ & | & \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} | \\ \mathbf{Y}_0 \\ | \end{bmatrix} \begin{bmatrix} \mathbf{X}_1 \end{bmatrix} \begin{bmatrix} -- & \mathbf{Y}_1 & -- \end{bmatrix}$$

where $\mathbf{X}_0, \mathbf{X}_1, \mathbf{Y}_0$ and $\mathbf{Y}_1$ are uniform matrices of appropriate dimensions.

9.2 Formal Proof of Baby Subspace Hiding Assumption

Given the above intuition, we now present the formal proof of the baby subspace hiding assumption. We state and prove the following lemma.

Lemma 9.1. *The baby subspace hiding assumption as in Definition 6.2 holds over the output ring R of a ring homomorphic synthesizer.*

We prove this lemma via a sequence of hybrid experiments $\mathcal{H}_0$ through $\mathcal{H}_z$, as described below, where $z = \text{poly}(\lambda)$ parameterizes the baby subspace hiding assumption (we also assume in the remainder of the description that $j^* \in [z]$ is the challenge-index chosen by the adversary $\mathcal{A}$ in the baby subspace hiding assumption experiment):

1. $\mathcal{H}_0$: This experiment is identical to the baby subspace hiding assumption experiment.

2. $\mathcal{H}_j$ for $j \in [j^* - 1]$: This experiment is identical to the previous experiment $\mathcal{H}_{j-1}$ except for the manner in which the challenger generates the guard matrices $\mathbf{R}_j$ and $\mathbf{L}_{j+1}$. Suppose that in the experiment $\mathcal{H}_{j-1}$, the guard matrices $\mathbf{R}_j$ and $\mathbf{L}_{j+1}$ are generated as:

$$\mathbf{R}_j = \begin{bmatrix} -- & \mathbf{R}_{j,1} & -- \\ -- & \mathbf{R}_{j,2} & -- \end{bmatrix} \quad , \quad \mathbf{L}_{j+1} = \begin{bmatrix} | & | \\ \mathbf{L}_{j+1,1} & \mathbf{L}_{j+1,2} \\ | & | \end{bmatrix} .$$

In the experiment $\mathcal{H}_j$, the submatrices $\mathbf{R}_{j,2}$ and $\mathbf{L}_{j+1,1}$ are replaced by uniformly random matrices. More concretely, in the experiment $\mathcal{H}_j$, we have

$$\mathbf{R}_j = \begin{bmatrix} -- & \mathbf{R}_{j,1} & -- \\ -- & \mathbf{X}_1 & -- \end{bmatrix} \quad , \quad \mathbf{L}_{j+1} = \begin{bmatrix} | & | \\ \mathbf{X}_2 & \mathbf{L}_{j+1,2} \\ | & | \end{bmatrix} ,$$

where $\mathbf{X}_1$ and $\mathbf{X}_2$ are matrices of the appropriate dimensions with uniformly sampled entries from the ring R.

3. $\mathcal{H}_j$ for $j \in [j^*, z-1]$: This experiment is identical to the previous experiment $\mathcal{H}_{j-1}$ except for the manner in which the challenger generates the guard matrices $\mathbf{R}_{z-(j-j^*)-1}$ and $\mathbf{L}_{z-(j-j^*)}$. Suppose that in the experiment $\mathcal{H}_{j-1}$, the guard matrices $\mathbf{R}_{z-(j-j^*)-1}$ and $\mathbf{L}_{z-(j-j^*)}$ are generated as:

$$\mathbf{R}_{z-(j-j^*)-1} = \begin{bmatrix} -- & \mathbf{R}_{z-(j-j^*)-1,1} & -- \\ -- & \mathbf{R}_{z-(j-j^*)-1,2} & -- \end{bmatrix} \quad , \quad \mathbf{L}_{z-(j-j^*)} = \begin{bmatrix} | & | \\ \mathbf{L}_{z-(j-j^*),1} & \mathbf{L}_{z-(j-j^*),2} \\ | & | \end{bmatrix} .$$

In the experiment $\mathcal{H}_j$, the submatrices $\mathbf{R}_{z-(j-j^*)-1,2}$ and $\mathbf{L}_{z-(j-j^*),1}$ are replaced by uniformly random matrices. More concretely, in the experiment $\mathcal{H}_j$, we have

$$\mathbf{R}_{z-(j-j^*)-1} = \begin{bmatrix} -- & \mathbf{Y}_1 & -- \\ -- & \mathbf{R}_{z-(j-j^*)-1,2} & -- \end{bmatrix} \quad , \quad \mathbf{L}_{z-(j-j^*)} = \begin{bmatrix} | & | \\ \mathbf{L}_{z-(j-j^*),1} & \mathbf{Y}_2 \\ | & | \end{bmatrix} ,$$

where $\mathbf{Y}_1$ and $\mathbf{Y}_2$ are matrices of the appropriate dimensions with uniformly sampled entries from the ring R.

4. $\mathcal{H}_z$: In this hybrid, the challenger does not generate the challenge element depending on the bit b. Instead, it generates a uniform square matrix $\mathbf{C}$ and provides it to the adversary $\mathcal{A}$, except when $j^* = 1$ or $j^* = z$, in which cases, the challenger provides the adversary with either $\mathbf{I}^\ell \mathbf{C}$, or $\mathbf{C}(\mathbf{I}^\ell)^T$, respectively.

Indistinguishability of Outer Hybrids. We state and prove the following lemma.

Lemma 9.2. *If R is the output of a ring homomorphic synthesizer, then hybrid $\mathcal{H}_0$ is computationally indistinguishable from hybrid $\mathcal{H}_1$.*

Proof. We begin by focusing on the guard submatrix $\mathbf{R}_{1,2}$. Note that the only terms involving $\mathbf{R}_{1,2}$ are the challenge-relevant terms, There are other terms that might reveal information on $\mathbf{R}_{1,2}$ (those terms that contain $\mathbf{L}_{2,1}$), but we shall ignore them for now. Let $\mathbf{H}_1$ be a challenge-relevant term of the form

$$\mathbf{H}_1 = \begin{bmatrix} \mathbf{X} & \mathbf{X} \end{bmatrix} \cdot \begin{bmatrix} -- & \mathbf{R}_{1,1} & -- \\ -- & \mathbf{R}_{1,2} & -- \end{bmatrix}$$

so if we can show that $\mathbf{H}_1$ is indistinguishable from random, then we are done.

In lemma 10.4, we show that for any random matrices $\mathbf{R}_{1,1} \in R^{n\times 2m}$, $\mathbf{R}_{1,2} \in R^{n\times 2m}$, $\mathbf{L}_{2,1} \in R^{2m\times n}$, and $\mathbf{L}_{2,2} \in R^{2m\times n}$, subject to the constraint that

$$\mathbf{R}_{1,1}\mathbf{L}_{2,2} = \mathbf{0}_R \quad \text{and} \quad \mathbf{R}_{1,2}\mathbf{L}_{2,1} = \mathbf{0}_R,$$

and uniformly random matrices $\mathbf{X}_1, \mathbf{X}_2 \in R^{n\times n}$ and $\mathbf{Z} \in R^{n\times 2m}$, then the following two tuples are indistinguishable:

$$(\mathbf{R}_{1,1}, \mathbf{R}_{1,2}, \mathbf{L}_{2,1}, \mathbf{L}_{2,2}, (\mathbf{X}_1\mathbf{R}_{1,1} + \mathbf{X}_2\mathbf{R}_{1,2})) \text{ and } (\mathbf{R}_{1,1}, \mathbf{R}_{1,2}, \mathbf{L}_{2,1}, \mathbf{L}_{2,2}, \mathbf{Z}) . \qquad \square$$

Note that $(\mathbf{X}_1\mathbf{R}_{1,1} + \mathbf{X}_2\mathbf{R}_{1,2})$ is the distribution of terms of the form $\mathbf{H}_1$ if $\mathbf{R}_{1,2}$ is distributed correctly, and $\mathbf{Z}$ is the distribution of terms of the form $\mathbf{H}_1$ if $\mathbf{R}_{1,2}$ is distributed randomly (note that we are using the fact that multiplication of a two random matrices is random, which follows from Theorem 4.2).

Given $\mathbf{R}_{1,1}$, $\mathbf{R}_{1,2}$, $\mathbf{L}_{2,1}$, $\mathbf{L}_{2,2}$ and terms of the form $\mathbf{H}_1$ that are either real or random, we can easily simulate all of the remaining terms with the appropriate distribution. Note that if we have randomized $\mathbf{R}_{1,2}$, then $\mathbf{L}_{2,1}$ is now completely independent of everything else and distributed randomly as desired. So, by invoking Lemma 10.4, we complete the proof of Lemma 9.2.

We also state the following lemma, the proof of which follows from arguments very similar to those in the proof of Lemma 9.2 and is hence not detailed.

Lemma 9.3. *If R is the output of a ring homomorphic synthesizer, then hybrid $\mathcal{H}_{z-2}$ is computationally indistinguishable from hybrid $\mathcal{H}_{z-1}$.*

Indistinguishability of Inner Hybrids. We now state and prove the following lemma.

Lemma 9.4. *If R is the output of a ring homomorphic synthesizer, then for each $j \in [2, j^* - 1]$, hybrid $\mathcal{H}_{j-1}$ is computationally indistinguishable from hybrid $\mathcal{H}_j$.*

Proof. As before, let's look closely at one guard submatrix in particular: namely, $\mathbf{R}_{j,2}$. Note that the only terms involving $\mathbf{R}_{j,2}$ are the challenge-relevant terms. There are other terms that might reveal information on $\mathbf{R}_{j,2}$ (those terms that contain $\mathbf{L}_{j+1,1}$), but we shall ignore them for now. Let $\mathbf{H}_j$ be a such a challenge-relevant term of the form

$$\mathbf{H}_j = \begin{bmatrix} | & | \\ \mathbf{L}_{j,1} & \mathbf{L}_{j,2} \\ | & | \end{bmatrix} \cdot \begin{bmatrix} \mathbf{X} & \mathbf{0}_R \\ \mathbf{0}_R & \mathbf{X} \end{bmatrix} \cdot \begin{bmatrix} -- & \mathbf{R}_{j,1} & -- \\ -- & \mathbf{R}_{j,2} & -- \end{bmatrix}$$

Now observe that in hybrid $\mathcal{H}_{j-1}$, the submatrix $\mathbf{L}_{j,1}$ has already been randomized (as well as its predecessor guard submatrix $\mathbf{R}_{j-1,1}$), so we may replace it with a random matrix $\mathbf{Z}$ without loss of generality. In other words, we have

$$\mathbf{H}_j = \begin{bmatrix} | & | \\ \mathbf{Z} & \mathbf{L}_{j,2} \\ | & | \end{bmatrix} \cdot \begin{bmatrix} \mathbf{X} & \mathbf{0}_R \\ \mathbf{0}_R & \mathbf{X} \end{bmatrix} \cdot \begin{bmatrix} -- & \mathbf{R}_{j,1} & -- \\ -- & \mathbf{R}_{j,2} & -- \end{bmatrix},$$

or equivalently,

$$\mathbf{H}_j = \mathbf{Z}\mathbf{X}_1\mathbf{R}_{j,1} + \mathbf{L}_{j,2}\mathbf{X}_2\mathbf{R}_{j,2},$$

where $\mathbf{X}_1$ and $\mathbf{X}_2$ are submatrices of appropriate dimensions with entries sampled uniformly from the ring R. Now when we additionally replace the submatrix $\mathbf{R}_{j,2}$ with a uniformly random submatrix $\mathbf{Z}'$ of the same dimension, the challenge-relevant term takes the form

$$\mathbf{H}_j = \mathbf{Z}\mathbf{X}_1\mathbf{R}_{j,1} + \mathbf{L}_{j,2}\mathbf{X}_2\mathbf{Z}'.$$

In Lemma 10.5, we show that for random matrices of appropriate dimensions $\mathbf{R}_{j-1,1} \in R^{n\times 2m}$, $\mathbf{L}_{j,2} \in R^{2m\times n}$, $\mathbf{R}_{j,1} \in R^{n\times 2m}$, $\mathbf{R}_{j,2} \in R^{n\times 2m}$, $\mathbf{L}_{j+1,1} \in R^{2m\times n}$, and $\mathbf{L}_{j+1,2} \in R^{2m\times n}$ subject to the constraints that,

$$\mathbf{R}_{j-1,1}\mathbf{L}_{j,2} = \mathbf{0}_R \quad \text{and} \quad \mathbf{R}_{j,1}\mathbf{L}_{j+1,2} = \mathbf{0}_R \quad \text{and} \quad \mathbf{R}_{j,2}\mathbf{L}_{j+1,1} = \mathbf{0}_R,$$

and for uniformly random matrices $\mathbf{X}_1, \mathbf{X}_2 \in R^{n\times n}$, $\mathbf{Z} \in R^{n\times 2m}$ and $\mathbf{Z}' \in R^{n\times 2m}$, given the tuple of matrices

$$(\mathbf{R}_{j-1,1}, \mathbf{Z}, \mathbf{L}_{j,2}, \mathbf{R}_{j,1}, \mathbf{R}_{j,2}, \mathbf{L}_{j+1,1}, \mathbf{L}_{j+1,2}),$$

the following terms are computationally indistinguishable:

$$(\mathbf{Z}\mathbf{X}_1\mathbf{R}_{j,1} + \mathbf{L}_{j,2}\mathbf{X}_2\mathbf{R}_{j,2}) \qquad \text{and} \qquad (\mathbf{Z}\mathbf{X}_1\mathbf{R}_{j,1} + \mathbf{L}_{j,2}\mathbf{X}_2\mathbf{Z}').$$ □

Note that on the left is the distribution of terms of the form $\mathbf{H}_j$ if $\mathbf{R}_{j,2}$ is distributed correctly (as in hybrid $\mathcal{H}_{j-1}$), and on the right is the distribution of terms of the form $\mathbf{H}_j$ if $\mathbf{R}_{j,2}$ is substituted with a uniformly random submatrix $\mathbf{Z}'$ (as in hybrid $\mathcal{H}_j$). Additionally, given the tuple of matrices as described above and terms of the form $\mathbf{H}_j$, we can easily simulate all of the remaining terms with the appropriate distribution. Finally, once we have randomized $\mathbf{R}_{j,2}$, then $\mathbf{L}_{j+1,1}$ is now completely independent of everything else and distributed randomly as desired. So, by invoking Lemma 10.5, we complete the proof of Lemma 9.4.

We also state the following lemma, the proof of which follows from arguments very similar to those in the proof of Lemma 9.4 and is hence not detailed.

Lemma 9.5. *If R is the output of a ring homomorphic synthesizer, then for each $j \in [j^* + 1, z - 1]$, hybrid $\mathcal{H}_{j-1}$ is computationally indistinguishable from hybrid $\mathcal{H}_j$.*

Indistinguishability of $\mathcal{H}_{z-1}$ and $\mathcal{H}_z$. Finally, we state and prove the following lemma.

Lemma 9.6. *If R is the output of a ring homomorphic synthesizer, then hybrid $\mathcal{H}_{z-1}$ is computationally indistinguishable from hybrid $\mathcal{H}_z$.*

Proof. Note that in both the hybrids $\mathcal{H}_{z-1}$ and $\mathcal{H}_z$, all guard matrices $\{\mathbf{L}_j, \mathbf{R}_j\}$ that could be partially randomized (without impacting the output of their desired product distributions as mandated by the subspace hiding the assumption) have, in fact, been partially randomized. The only difference between the hybrids $\mathcal{H}_{z-1}$ and $\mathcal{H}_z$ is that the challenge element to be provided to the adversary $\mathcal{A}$ is additionally randomized in the final hybrid $\mathcal{H}_z$.

Recall that, in $\mathcal{H}_{z-1}$ (and all the hybrids before it), the challenge elements have one of the two forms stated below (depending on the challenge bit b):

$$\mathbf{C}_0 = \begin{bmatrix} | & | \\ \mathbf{L}_{j^*,1} & \mathbf{L}_{j^*,2} \\ | & | \end{bmatrix} \cdot \begin{bmatrix} \mathbf{X} & \mathbf{0}_R \\ \mathbf{0}_R & \mathbf{0}_R \end{bmatrix} \cdot \begin{bmatrix} -- & \mathbf{R}_{j^*,1} & -- \\ -- & \mathbf{R}_{j^*,2} & -- \end{bmatrix}$$

or

$$\mathbf{C}_1 = \begin{bmatrix} | & | \\ \mathbf{L}_{j^*,1} & \mathbf{L}_{j^*,2} \\ | & | \end{bmatrix} \cdot \begin{bmatrix} \mathbf{X} & \mathbf{0}_R \\ \mathbf{0}_R & \mathbf{X} \end{bmatrix} \cdot \begin{bmatrix} -- & \mathbf{R}_{j^*,1} & -- \\ -- & \mathbf{R}_{j^*,2} & -- \end{bmatrix}$$

Next, observe the following:

- The guard submatrix $\mathbf{L}_{j^*,1}$ is randomized in hybrid $\mathcal{H}_{j^*-1}$ and all subsequent hybrids.
- The guard submatrix $\mathbf{R}_{j^*,1}$ is randomized in hybrid $\mathcal{H}_{z-1}$ and all subsequent hybrids.

In other words, in $\mathcal{H}_{z-1}$, the challenge elements have one of the two forms stated below (depending on the challenge bit b):

$$\mathbf{C}_0 = \begin{bmatrix} | & | \\ \mathbf{X}_{j^*,1} & \mathbf{L}_{j^*,2} \\ | & | \end{bmatrix} \cdot \begin{bmatrix} \mathbf{X} & \mathbf{0}_R \\ \mathbf{0}_R & \mathbf{0}_R \end{bmatrix} \cdot \begin{bmatrix} -- & \mathbf{Y}_{j^*,1} & -- \\ -- & \mathbf{R}_{j^*,2} & -- \end{bmatrix}$$

or

$$\mathbf{C}_1 = \begin{bmatrix} | & | \\ \mathbf{X}_{j^*,1} & \mathbf{L}_{j^*,2} \\ | & | \end{bmatrix} \cdot \begin{bmatrix} \mathbf{X} & \mathbf{0}_R \\ \mathbf{0}_R & \mathbf{X} \end{bmatrix} \cdot \begin{bmatrix} -- & \mathbf{Y}_{j^*,1} & -- \\ -- & \mathbf{R}_{j^*,2} & -- \end{bmatrix},$$

or equivalently,

$$\mathbf{C}_0 = \mathbf{X}_{j^*,1}\mathbf{X}_1\mathbf{Y}_{j^*,1} \quad \text{or} \quad \mathbf{C}_1 = \mathbf{X}_{j^*,1}\mathbf{X}_1\mathbf{Y}_{j^*,1} + \mathbf{L}_{j^*,2}\mathbf{X}_2\mathbf{R}_{j^*,2},$$

where $\mathbf{X}_{j^*,1}$, $\mathbf{Y}_{j^*,1}$, $\mathbf{X}_1$ and $\mathbf{X}_2$ are uniformly random matrices. Now, observe that terms of the form

$$\mathbf{X}_{j^*,1}\mathbf{X}_1\mathbf{Y}_{j^*,1},$$

are computationally indistinguishable from random by Theorem 4.2 whenever $\mathbf{X}_1$ is uniformly random. This in turn implies that the distribution of the challenge element in hybrid $\mathcal{H}_{z-1}$ must be computationally indistinguishable from random (independent of the challenge bit b), and hence, computationally indistinguishable from that in hybrid $\mathcal{H}_z$ (independent of the challenge bit b). This completes the proof of Lemma 9.6. □

Putting Everything Together. Finally, the proof of Lemma 9.1 follows by putting together, in sequence, the proofs of Lemma 9.2, Lemma 9.4, Lemma 9.5, Lemma 9.3 and Lemma 9.6. This completes the proof of security of the baby subspace hiding assumption as in Definition 6.2.

10 Some Useful Lemmata

Lemma 10.1. *Let R be a finite ring, and let m, n be integers such that $m = 2n$ and $n > 3\log|R|$. Assume that $\mathbf{v} \leftarrow R^m$, and $\mathbf{v}^*$ sampled uniformly conditioned on $\mathbf{v}^t\mathbf{v}^* = 0_R$. If $\mathbf{a} \leftarrow R^n$, $\vec{b} \leftarrow R^n$, and $\mathbf{S} \leftarrow R^{n\times n}$ then we have*

$$(\mathbf{v}, \mathbf{v}^*) \overset{s}{\approx} (\mathbf{w}, \mathbf{w}^*),$$

where $\mathbf{w}^t = (\mathbf{a}^t, \mathbf{a}^t\mathbf{S}) \in R^m$ *and* $\mathbf{w}^* = \begin{bmatrix} -\mathbf{Sb} \\ \mathbf{b} \end{bmatrix} \in R^m$.

Proof. We use the notation $\mathbf{v} = (\mathbf{v}_1, \mathbf{v}_2)$ and $\mathbf{v}^* = (\mathbf{v}_1^*, \mathbf{v}_2^*)$ to denote the first and second half of $\mathbf{v}$ and $\mathbf{v}^*$, respectively. First, by applying Lemma 3.7 we know that $(\mathbf{v}_1^t, \mathbf{v}_2^t) \overset{s}{\approx} (\mathbf{a}^t, \mathbf{a}^t\mathbf{X})$. Now, in order to uniformly sample $\mathbf{w}^*$ conditioned on

$$\mathbf{a}^t\mathbf{w}_1^* + \mathbf{a}^t\mathbf{S}\mathbf{w}_2^* = 0,$$

we can first sample a uniform $\mathbf{b} := \mathbf{w}_2^* \leftarrow R^n$, and then sample $\mathbf{w}_1^*$ condition on the equation above. By rearranging, it follows that

$$\mathbf{a}^t\mathbf{w}_1^* = -\mathbf{a}^t\mathbf{Sb}.$$

Now observe that given all the terms in the equation above except $\mathbf{w}_1^*$, the distribution of $\mathbf{w}_1^*$ is $\mathbf{Sb} + \mathbf{k}$, where $\mathbf{Sb}$ is distributed as mentioned above and $\mathbf{k}$ is a uniform vector from the set $\mathsf{RKer}(\mathbf{a})$.[1] By applying Lemma 3.7 again, it follows that (with overwhemling probability) for every $\mathbf{k} \in \mathsf{RKer}(\mathbf{a})$ there exists a matrix $\mathbf{S}' \in R^{n\times n}$ such that $\mathbf{S}'\mathbf{y} = \mathbf{k}$. Therefore, we can say that the distribution of $\mathbf{w}_1^*$ is $\mathbf{Sb} + \mathbf{S}'\mathbf{b} = (\mathbf{S} + \mathbf{S}')\mathbf{b}$, where the distribution of $\mathbf{S}'$ is induced by $\mathsf{RKer}(\mathbf{a})$. Since $\mathbf{S}$ is uniform and independent of $\mathbf{S}'$, it follows that the distribution of $\mathbf{w}_1^*$ is statistically close to $\mathbf{Sy}$ where $\mathbf{S}$ is uniformly distributed. It follows that

$$(\mathbf{v}^t, \mathbf{v}^*) \overset{s}{\approx} ((\mathbf{a}^t, \mathbf{a}^t\mathbf{S}), \begin{bmatrix} -\mathbf{Sb} \\ \mathbf{b} \end{bmatrix}),$$

and hence

$$(\mathbf{v}, \mathbf{v}^*) \overset{s}{\approx} (\mathbf{w}, \mathbf{w}^*),$$

as required. □

[1] Recall that $\mathsf{RKer}(\mathbf{a})$ is the set of all vectors $\mathbf{t}$ such that $\mathbf{a}^t\mathbf{t} = 0_R$.

Corollary 10.2. *Let R be a finite ring, and let m, n, ℓ be integers such that $m = 2n$, $n > 3\ell \log |R|$, and m is an integer multiple of ℓ. Assume that $\mathbf{V} \leftarrow R^{\ell\times m}$, and $\mathbf{V}^* \in R^{m\times\ell}$ sampled uniformly conditioned on $\mathbf{V}\mathbf{V}^* = \mathbf{0}$. If $\mathbf{A} \leftarrow R^{\ell\times n}$, $\mathbf{B} \leftarrow R^{n\times\ell}$, and $\mathbf{S} \leftarrow R^{n\times n}$ then we have*

$$(\mathbf{V}, \mathbf{V}^*) \overset{s}{\approx} (\mathbf{W}, \mathbf{W}^*),$$

where $\mathbf{W} = [\mathbf{A} \mid \mathbf{AS}] \in R^{\ell\times m}$ *and* $\mathbf{W}^* = \begin{bmatrix} -\mathbf{SB} \\ \mathbf{B} \end{bmatrix} \in R^{m\times\ell}$.

Proof. It follows by applying the previous lemma, and observing the fact that $M_\ell(R)$ (the set of square $\ell \times \ell$ matrices over R) forms a finite ring. □

We also need the following corollary, which plays an essential role in proving that "guard matrices" are distributed properly.

Corollary 10.3. *Let R be a finite ring, and let m, n, ℓ be integers such that $m = 2n$, $n > 3\ell \log |R|$, and m is an integer multiple of ℓ. Assume that $\mathbf{V} \leftarrow R^{\ell\times m}$, and $\mathbf{V}^* \in R^{m\times\ell}$ sampled uniformly conditioned on $\mathbf{V}\mathbf{V}^* = \mathbf{MB}$ where $\mathbf{M} \in R^{\ell\times m}$ is an arbitrary matrix and $\mathbf{B} \leftarrow R^{m\times\ell}$ is uniformly generated. If $\mathbf{A} \leftarrow R^{\ell\times n}$, $\mathbf{B} \leftarrow R^{n\times\ell}$, and $\mathbf{S} \leftarrow R^{n\times n}$ then we have*

$$(\mathbf{V}, \mathbf{V}^*) \overset{s}{\approx} (\mathbf{W}, \mathbf{W}^*),$$

where $\mathbf{W} = [\mathbf{A} \mid \mathbf{AS} + \mathbf{M}] \in R^{\ell\times m}$ *and* $\mathbf{W}^* = \begin{bmatrix} -\mathbf{SB} \\ \mathbf{B} \end{bmatrix} \in R^{m\times\ell}$.

Proof. This proof follows by the applying the previous corollary to the coset that is induced by the element $\mathbf{MB}$. (One can view the previous corollary as a special case of this one by setting $\mathbf{M}$ to be all-zero matrix.) □

Lemma 10.4. *Let m and n be integers such that $m \geq 6n \log |R|$ where R is the output space of a ring-embedded homomorphic synthesizer. Let the matrices $\mathbf{R}_1 \in R^{n\times 2m}$, and $\mathbf{R}_2 \in R^{n\times 2m}$ $\mathbf{L}_1 \in R^{2m\times n}$, $\mathbf{L}_2 \in R^{2m\times n}$, be random subject to the constraint that*

$$\mathbf{R}_1\mathbf{L}_2 = 0 \qquad \textit{and} \qquad \mathbf{R}_2\mathbf{L}_1 = 0$$

Suppose we sample two matrices $\mathbf{X}_1, \mathbf{X}_2 \in R^{n\times n}$ uniformly at random, and let $\mathbf{Z} \in R^{n\times 2m}$ be sampled uniformly at random as well. We claim the following two tuples are indistinguishable:

$$(\mathbf{R}_1, \mathbf{R}_2, \mathbf{L}_1, \mathbf{L}_2, (\mathbf{X}_1\mathbf{R}_1 + \mathbf{X}_2\mathbf{R}_2)) \ \textit{and}\ (\mathbf{R}_1, \mathbf{R}_2, \mathbf{L}_1, \mathbf{L}_2, \mathbf{Z})$$

Proof. To provide a better intuition, let's start by visualizing what we want to prove. Suppose we have the following matrices:

$$\begin{bmatrix} -- & \mathbf{R}_1 & -- \end{bmatrix}, \begin{bmatrix} -- & \mathbf{R}_2 & -- \end{bmatrix}, \begin{bmatrix} | \\ \mathbf{L}_1 \\ | \end{bmatrix}, \begin{bmatrix} | \\ \mathbf{L}_2 \\ | \end{bmatrix}$$

We claim that, given two sets of orthogonal matrices of ring elements: $\mathbf{R}_1$, $\mathbf{L}_2$ and $\mathbf{R}_2$, $\mathbf{L}_1$, the following is indistinguishable from random for random $\mathbf{X}_1$, $\mathbf{X}_2$:

$$\begin{bmatrix} \mathbf{X}_1 & \mathbf{X}_2 \end{bmatrix} \begin{bmatrix} -- & \mathbf{R}_1 & -- \\ -- & \mathbf{R}_2 & -- \end{bmatrix}$$

By Corollary 10.2, for random $\mathbf{F} \in R^{m\times m}$, $\mathbf{G} \in R^{m\times n}$, and $\mathbf{H} \in R^{n\times m}$, the set of matrices

$$\mathbf{A} \in R^{n\times 2m} := \begin{bmatrix} \mathbf{G} & \mathbf{GF} \end{bmatrix}, \qquad \mathbf{B} \in R^{2m\times n} := \begin{bmatrix} -\mathbf{FH} \\ \mathbf{H} \end{bmatrix}$$

are statistically indistinguishable from a randomly chosen pair of matrices

$$\left(\mathbf{A}' \in R^{2m\times n}, \mathbf{B}' \in R^{n\times 2m}\right)$$

such that $\mathbf{AB} = 0$. Given this, we can replace the matrices in our assumption with ones of this form.

Suppose we define uniformly random matrices $\widetilde{\mathbf{R}_1} \in R^{n\times m}$, $\widetilde{\mathbf{R}_2} \in R^{n\times m}$, $\widetilde{\mathbf{L}_1} \in R^{m\times n}$, $\widetilde{\mathbf{L}_2} \in R^{m\times n}$, $\mathbf{U} \in R^{m\times m}$, and $\mathbf{V} \in R^{m\times m}$ and set:

$$\mathbf{R}_1 = \begin{bmatrix}\widetilde{\mathbf{R}_1} & \widetilde{\mathbf{R}_1}\mathbf{U}\end{bmatrix}, \quad \mathbf{R}_2 = \begin{bmatrix}\widetilde{\mathbf{R}_1} & \widetilde{\mathbf{R}_2}\mathbf{V}\end{bmatrix}, \quad \mathbf{L}_1 = \begin{bmatrix}-\mathbf{V}\widetilde{\mathbf{L}_1} \\ \widetilde{\mathbf{L}_1}\end{bmatrix}, \quad \mathbf{L}_2 = \begin{bmatrix}-\mathbf{U}\widetilde{\mathbf{L}_2} \\ \widetilde{\mathbf{L}_2}\end{bmatrix}$$

Then our lemma is exactly equivalent to distinguishing the following:

$$\left(\left[\widetilde{\mathbf{R}_1}, \widetilde{\mathbf{R}_1}\mathbf{U}\right], \left[\widetilde{\mathbf{R}_2}, \widetilde{\mathbf{R}_2}\mathbf{V}\right], \left[-\widetilde{\mathbf{L}_1}\mathbf{V}, \widetilde{\mathbf{L}_1}\right], \left[-\mathbf{U}\widetilde{\mathbf{L}_2}, \widetilde{\mathbf{L}_2}\right], (\mathbf{X}_1\mathbf{R}_1 + \mathbf{X}_2\mathbf{R}_2)\right) \text{ and}$$

$$\left(\left[\widetilde{\mathbf{R}_1}, \widetilde{\mathbf{R}_1}\mathbf{U}\right], \left[\widetilde{\mathbf{R}_2}, \widetilde{\mathbf{R}_2}\mathbf{V}\right], \left[-\widetilde{\mathbf{L}_1}\mathbf{V}, \widetilde{\mathbf{L}_1}\right], \left[-\mathbf{U}\widetilde{\mathbf{L}_2}, \widetilde{\mathbf{L}_2}\right], \mathbf{Z}\right)$$

Now let's work graphically. Note that

$$\begin{bmatrix}\mathbf{X}_1 & \mathbf{X}_2\end{bmatrix}\begin{bmatrix}-- & \mathbf{R}_1 & -- \\ -- & \mathbf{R}_2 & --\end{bmatrix} = \begin{bmatrix}\mathbf{X}_1 & \mathbf{X}_2\end{bmatrix}\begin{bmatrix}\widetilde{\mathbf{R}_1} & \mathbf{U}\widetilde{\mathbf{R}_1} \\ \widetilde{\mathbf{R}_2} & \mathbf{V}\widetilde{\mathbf{R}_2}\end{bmatrix} =$$

$$\begin{bmatrix}\mathbf{X}_1\widetilde{\mathbf{R}_1} + \mathbf{X}_2\widetilde{\mathbf{R}_2} || & \mathbf{X}_1\widetilde{\mathbf{R}_1}\mathbf{U} + \mathbf{X}_2\widetilde{\mathbf{R}_2}\mathbf{V}\end{bmatrix}$$

Let $\mathbf{C} \in R^{n\times m}$ and $\mathbf{D} \in R^{n\times m}$ be uniformly random matrices. By Theorem 4.2, we know that the following tuples are indistinguishable:

$$\left(\widetilde{\mathbf{R}_1}, \mathbf{X}_1\widetilde{\mathbf{R}_1}\right) \quad \text{and} \quad \left(\widetilde{\mathbf{R}_1}, \mathbf{C}\right)$$

$$\left(\widetilde{\mathbf{R}_2}, \mathbf{X}_2\widetilde{\mathbf{R}_2}\right) \quad \text{and} \quad \left(\widetilde{\mathbf{R}_2}, \mathbf{C}\right)$$

Using this indistinguishability, we can reduce the output of our assumption to

$$\begin{bmatrix}\mathbf{C} + \mathbf{D} & ||\mathbf{CU} + \mathbf{DV}\end{bmatrix}$$

Note that

$$\begin{bmatrix}\mathbf{C} + \mathbf{D} & ||\mathbf{CU} + \mathbf{DV}\end{bmatrix} = \begin{bmatrix}\mathbf{C} + \mathbf{D} & ||\left(\mathbf{C} + \mathbf{D}\right)\mathbf{U} + \mathbf{D}\left(\mathbf{V} - \mathbf{U}\right)\end{bmatrix}$$

Since $\mathbf{C}$ and $\mathbf{D}$ are uniformly random, we have that, for some random $\mathbf{E} \in R^{n\times m}$, the above is identically distributed to

$$\begin{bmatrix}\mathbf{E} & |\mathbf{EU} + \mathbf{D}\left(\mathbf{V} - \mathbf{U}\right)\end{bmatrix}$$

Since $\mathbf{V}$ and $\mathbf{U}$ are random, we know that, for some random $\mathbf{W} \in R^{n\times m}$, the above is identically distributed to

$$\begin{bmatrix}\mathbf{E} & |\mathbf{EU} + \mathbf{DW}\end{bmatrix}$$

We note that this is the case even if $\mathbf{V}$ and $\mathbf{U}$ are publicly known. Let $\mathbf{K} \in R^{n\times m}$ be a uniformly random matrix, by Theorem 4.2 it is easy to see that the following are indistinguishable from random

$$(\mathbf{W}, \mathbf{DW}) \quad \text{and} \quad (\mathbf{W}, \mathbf{K})$$

This allows us to randomize the second term in the above matrix, completing the proof. □

Lemma 10.5. *Let m and n be integers such that $m \geq 6n\log|R|$ where R is the output space of a ring-embedded homomorphic synthesizer. Let the matrices $\mathbf{R}_{1,2} \in R^{n\times 2m}$, $\mathbf{L}_{2,1} \in R^{2m\times n}$, $\mathbf{R}_{2,1} \in R^{n\times 2m}$, and $\mathbf{R}_{2,2} \in R^{n\times 2m}$, $\mathbf{L}_{3,1} \in R^{2m\times n}$, and $\mathbf{L}_{3,2} \in R^{2m\times n}$ be random subject to the constraint that, for all relevant i:*

$$\mathbf{L}_{i,0}\mathbf{R}_{i+1,1} = 0 \qquad \textit{and} \qquad \mathbf{L}_{i,1}\mathbf{R}_{i+1,0} = 0$$

Suppose we sample two matrices $\mathbf{X}_1, \mathbf{X}_2 \in R^{n\times n}$ *uniformly at random, and let* $\mathbf{Z} \in R^{n\times 2m}$ *and* $\mathbf{Z}' \in R^{n\times 2m}$ *be sampled uniformly at random as well. We claim the following: given the terms*

$$\mathbf{R}_{1,2}, \mathbf{L}_{2,1}, \mathbf{R}_{2,1}, \mathbf{R}_{2,2}, \mathbf{L}_{3,1}, \mathbf{L}_{3,2}, \mathbf{Z}$$

the following tuples are indistinguishable:

$$(\mathbf{L}_{2,1}\mathbf{X}_1\mathbf{R}_{2,1} + \mathbf{Z}\mathbf{X}_2\mathbf{R}_{2,2}) \quad \textit{and} \ (\mathbf{L}_{2,1}\mathbf{X}_1\mathbf{Z}' + \mathbf{Z}\mathbf{X}_2\mathbf{R}_{2,2})$$

Proof. Observe that essentially what we are trying to prove is that the following two products are indistinguishable from each other:

$$\begin{bmatrix} | & | \\ \mathbf{L}_{2,1} & \mathbf{Z} \\ | & | \end{bmatrix} \begin{bmatrix} \mathbf{X}_1 & 0 \\ 0 & \mathbf{X}_2 \end{bmatrix} \begin{bmatrix} -- & \mathbf{R}_{2,1} & -- \\ -- & \mathbf{R}_{2,2} & -- \end{bmatrix}$$

and

$$\begin{bmatrix} | & | \\ \mathbf{L}_{2,1} & \mathbf{Z} \\ | & | \end{bmatrix} \begin{bmatrix} \mathbf{X}_1 & 0 \\ 0 & \mathbf{X}_2 \end{bmatrix} \begin{bmatrix} -- & \mathbf{Z}' & -- \\ -- & \mathbf{R}_{2,2} & -- \end{bmatrix}$$

even when given all of the matrices in the assumption. Note that we are giving out all of the "guard" matrices in the "real" product–$\mathbf{L}_{2,1}$, $\mathbf{Z}$, $\mathbf{R}_{2,1}$ and $\mathbf{R}_{2,2}$–as well as matrices $\mathbf{R}_{1,2}$, $\mathbf{L}_{3,2}$, and $\mathbf{L}_{3,1}$, which are orthogonal to $\mathbf{L}_{2,1}$, $\mathbf{R}_{2,1}$, and $\mathbf{R}_{2,2}$, respectively. Note that $\mathbf{Z}$ is totally random and that it is completely independent from all other terms (we are not giving out anything orthogonal to it). This will be crucial for our proofs.

By Corollary 10.2, for random $\mathbf{F} \in R^{m\times m}$, $\mathbf{G} \in R^{m\times n}$, and $\mathbf{H} \in R^{n\times m}$, the set of matrices

$$\mathbf{A} \in R^{n\times 2m} := \begin{bmatrix} \mathbf{G} & \mathbf{GF} \end{bmatrix}, \qquad \mathbf{B} \in R^{2m\times n} := \begin{bmatrix} -\mathbf{FH} \\ \mathbf{H} \end{bmatrix}$$

are statistically indistinguishable from a randomly chosen pair of matrices

$$\left(\mathbf{A}' \in R^{2m\times n}, \mathbf{B}' \in R^{n\times 2m}\right)$$

such that $\mathbf{AB} = 0$. Given this, once again we can replace the matrices in our assumption with ones of this form.

As before, we define many tuples of orthogonal matrices Let $\widetilde{\mathbf{R}_{1,2}} \in R^{m\times n}$, $\widetilde{\mathbf{L}_{2,1}} \in R^{n\times m}$, $\widetilde{\mathbf{R}_{2,1}} \in R^{m\times n}$, $\widetilde{\mathbf{L}_{3,2}} \in R^{n\times m}$,$\widetilde{\mathbf{R}_{2,2}} \in R^{m\times n}$, $\widetilde{\mathbf{L}_{3,1}} \in R^{n\times m}$, $\mathbf{V}_1 \in R^{m\times m}$, $\mathbf{U}_2 \in R^{m\times m}$, and $\mathbf{V}_2 \in R^{m\times m}$ be distributed uniformly at random. We can define our original matrices in the following way:

$$\mathbf{R}_{1,2} = \begin{bmatrix} \widetilde{\mathbf{R}_{1,2}} & \widetilde{\mathbf{R}_{1,2}}\mathbf{V}_1 \end{bmatrix}, \quad \mathbf{L}_{2,1} = \begin{bmatrix} -\widetilde{\mathbf{V}_1\mathbf{L}_{2,1}} \\ \widetilde{\mathbf{L}_{2,1}} \end{bmatrix}, \quad \mathbf{R}_{2,1} = \begin{bmatrix} \widetilde{\mathbf{R}_{2,1}} & \widetilde{\mathbf{R}_{2,1}}\mathbf{U}_2 \end{bmatrix}, \quad \mathbf{L}_{3,2} = \begin{bmatrix} -\mathbf{U}_2\widetilde{\mathbf{L}_{3,2}} \\ \widetilde{\mathbf{L}_{3,2}} \end{bmatrix},$$

$$\mathbf{R}_{2,2} = \begin{bmatrix} \widetilde{\mathbf{R}_{2,2}} & \widetilde{\mathbf{R}_{2,2}}\mathbf{V}_2 \end{bmatrix}, \quad \mathbf{L}_{3,1} = \begin{bmatrix} -\mathbf{V}_2\widetilde{\mathbf{L}_{3,1}} \\ \widetilde{\mathbf{L}_{3,1}} \end{bmatrix}$$

As a simplification, suppose we set $\mathbf{Z} = [\mathbf{Z}_1 || \mathbf{Z}_2]$ for $\mathbf{Z}_1, \mathbf{Z}_2 \in R^{m\times n}$. We can rewrite

$$\begin{bmatrix} | & | \\ \mathbf{L}_{2,1} & \mathbf{Z} \\ | & | \end{bmatrix} \begin{bmatrix} \mathbf{X}_1 & 0 \\ 0 & \mathbf{X}_2 \end{bmatrix} \begin{bmatrix} -- & \mathbf{R}_{2,1} & -- \\ -- & \mathbf{R}_{2,2} & -- \end{bmatrix}$$

as

$$\begin{bmatrix} | & | \\ \mathbf{V}_1\widetilde{\mathbf{L}_{2,1}} & \mathbf{Z}_1 \\ | & | \\ | & | \\ \widetilde{\mathbf{L}_{2,1}} & \mathbf{Z}_2 \\ | & | \end{bmatrix} \begin{bmatrix} \mathbf{X}_1 & 0 \\ 0 & \mathbf{X}_2 \end{bmatrix} \begin{bmatrix} -- & \widetilde{\mathbf{R}_{2,1}} & -- & -- & \widetilde{\mathbf{R}_{2,1}}\mathbf{U}_2 & -- \\ -- & \widetilde{\mathbf{R}_{2,2}} & -- & -- & \widetilde{\mathbf{R}_{2,2}}\mathbf{V}_2 & -- \end{bmatrix}$$

$$= \begin{bmatrix} | & | \\ \mathbf{V}_1\widetilde{\mathbf{L}_{2,1}}\mathbf{X}_1 & \mathbf{Z}_1\mathbf{X}_2 \\ | & | \\ | & | \\ \widetilde{\mathbf{L}_{2,1}}\mathbf{X}_1 & \mathbf{Z}_2\mathbf{X}_2 \\ | & | \end{bmatrix} \begin{bmatrix} -- & \widetilde{\mathbf{R}_{2,1}} & -- & -- & \widetilde{\mathbf{R}_{2,1}}\mathbf{U}_2 & -- \\ -- & \widetilde{\mathbf{R}_{2,2}} & -- & -- & \widetilde{\mathbf{R}_{2,2}}\mathbf{V}_2 & -- \end{bmatrix}$$

Let $\mathbf{C}_1 \in R^{m\times n}$, $\mathbf{C}_2 \in R^{m\times n}$, and $\mathbf{C}_3 \in R^{m\times n}$ be uniformly random matrices. By Theorem 4.2, we know that the following are indistinguishable from random:

$$\left(\widetilde{\mathbf{L}_{2,1}}, \widetilde{\mathbf{L}_{2,1}}\right)\mathbf{X}_1 \quad \text{and} \quad \left(\widetilde{\mathbf{L}_{2,1}}, \mathbf{C}_1\right)$$

$$(\mathbf{Z}, \mathbf{Z}\mathbf{X}_2) \quad \text{and} \quad \left(\mathbf{Z}, [\mathbf{C}_2||\mathbf{C}_3]^T\right)$$

We can rewrite the above product as

$$\begin{bmatrix} | & | \\ \mathbf{V}_1\mathbf{C}_1 & \mathbf{C}_2 \\ | & | \\ | & | \\ \mathbf{C}_1 & \mathbf{C}_3 \\ | & | \end{bmatrix} \begin{bmatrix} -- & \widetilde{\mathbf{R}_{2,1}} & -- & -- & \widetilde{\mathbf{R}_{2,1}}\mathbf{U}_2 & -- \\ -- & \widetilde{\mathbf{R}_{2,2}} & -- & -- & \widetilde{\mathbf{R}_{2,2}}\mathbf{V}_2 & -- \end{bmatrix}$$

If we expand this matrix, we get

$$\begin{bmatrix} \mathbf{V}_1\mathbf{C}_1\widetilde{\mathbf{R}_{2,1}} + \mathbf{C}_2\widetilde{\mathbf{R}_{2,2}} & \mathbf{V}_1\mathbf{C}_1\widetilde{\mathbf{R}_{2,1}}\mathbf{U}_2 + \mathbf{C}_2\widetilde{\mathbf{R}_{2,2}}\mathbf{V}_2 \\ \mathbf{C}_1\widetilde{\mathbf{R}_{2,1}} + \mathbf{C}_3\widetilde{\mathbf{R}_{2,2}} & \mathbf{C}_1\widetilde{\mathbf{R}_{2,1}}\mathbf{U}_2 + \mathbf{C}_3\widetilde{\mathbf{R}_{2,2}}\mathbf{V}_2 \end{bmatrix}$$

Suppose we also assume the adversary knows $\mathbf{V}_1$, $\mathbf{U}_2$, and $\mathbf{V}_2$. Recall that matrices of the form

$$\begin{bmatrix} \mathbf{I} & \mathbf{X} \\ 0 & \mathbf{I} \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} \mathbf{Y} & \mathbf{I} \\ \mathbf{I} & 0 \end{bmatrix}$$

are invertible for any ring and for any $\mathbf{X}, \mathbf{Y}$. Now let's consider the product

$$\begin{bmatrix} \mathbf{I} & -\mathbf{V}_1 \\ 0 & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{V}_1\mathbf{C}_1\widetilde{\mathbf{R}_{2,1}} + \mathbf{C}_2\widetilde{\mathbf{R}_{2,2}} & \mathbf{V}_1\mathbf{C}_1\widetilde{\mathbf{R}_{2,1}}\mathbf{U}_2 + \mathbf{C}_2\widetilde{\mathbf{R}_{2,2}}\mathbf{V}_2 \\ \mathbf{C}_1\widetilde{\mathbf{R}_{2,1}} + \mathbf{C}_3\widetilde{\mathbf{R}_{2,2}} & \mathbf{C}_1\widetilde{\mathbf{R}_{2,1}}\mathbf{U}_2 + \mathbf{C}_3\widetilde{\mathbf{R}_{2,2}}\mathbf{V}_2 \end{bmatrix} \begin{bmatrix} \mathbf{I} & 0 \\ -\mathbf{V}_2 & \mathbf{I} \end{bmatrix}$$

If we can show that certain terms in this product are random, then it immediately follows that the terms in our original matrix are indistinguishable from random since the operations are invertible and we are assuming that $\mathbf{V}_1$ and $\mathbf{V}_2$ are known to the adversary. Now, our goal is to show that $\mathbf{U}_2$ is indistinguishable from random. By working some calculation, we get

$$\begin{bmatrix} \mathbf{I} & -\mathbf{V}_1 \\ 0 & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{V}_1\mathbf{C}_1\widetilde{\mathbf{R}_{2,1}} + \mathbf{C}_2\widetilde{\mathbf{R}_{2,2}} & \mathbf{V}_1\mathbf{C}_1\widetilde{\mathbf{R}_{2,1}}\mathbf{U}_2 + \mathbf{C}_2\widetilde{\mathbf{R}_{2,2}}\mathbf{V}_2 \\ \mathbf{C}_1\widetilde{\mathbf{R}_{2,1}} + \mathbf{C}_3\widetilde{\mathbf{R}_{2,2}} & \mathbf{C}_1\widetilde{\mathbf{R}_{2,1}}\mathbf{U}_2 + \mathbf{C}_3\widetilde{\mathbf{R}_{2,2}}\mathbf{V}_2 \end{bmatrix} =$$

$$\begin{bmatrix} \mathbf{C}_2\widetilde{\mathbf{R}_{2,2}} - \mathbf{V}_1\mathbf{C}_3\widetilde{\mathbf{R}_{2,2}} & \mathbf{C}_2\widetilde{\mathbf{R}_{2,2}}\mathbf{V}_2 - \mathbf{V}_1\mathbf{C}_3\widetilde{\mathbf{R}_{2,2}}\mathbf{V}_2 \\ \mathbf{C}_1\widetilde{\mathbf{R}_{2,1}} + \mathbf{C}_3\widetilde{\mathbf{R}_{2,2}} & \mathbf{C}_1\widetilde{\mathbf{R}_{2,1}}\mathbf{U}_2 + \mathbf{C}_3\widetilde{\mathbf{R}_{2,2}}\mathbf{V}_2 \end{bmatrix}$$

and also that

$$\begin{bmatrix} \mathbf{C}_2\widetilde{\mathbf{R}_{2,2}} - \mathbf{V}_1\mathbf{C}_3\widetilde{\mathbf{R}_{2,2}} & \mathbf{C}_2\widetilde{\mathbf{R}_{2,2}}\mathbf{V}_2 - \mathbf{V}_1\mathbf{C}_3\widetilde{\mathbf{R}_{2,2}}\mathbf{V}_2 \\ \mathbf{C}_1\widetilde{\mathbf{R}_{2,1}} + \mathbf{C}_3\widetilde{\mathbf{R}_{2,2}} & \mathbf{C}_1\widetilde{\mathbf{R}_{2,1}}\mathbf{U}_2 + \mathbf{C}_3\widetilde{\mathbf{R}_{2,2}}\mathbf{V}_2 \end{bmatrix} \begin{bmatrix} -\mathbf{V}_2 & \mathbf{I} \\ \mathbf{I} & 0 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & \mathbf{C}_2\widetilde{\mathbf{R}_{2,2}} - \mathbf{V}_1\mathbf{C}_3\widetilde{\mathbf{R}_{2,2}} \\ \mathbf{C}_1\widetilde{\mathbf{R}_{2,1}}\mathbf{U}_2 + \mathbf{C}_3\widetilde{\mathbf{R}_{2,2}}\mathbf{V}_2 & \mathbf{C}_1\widetilde{\mathbf{R}_{2,1}} + \mathbf{C}_3\widetilde{\mathbf{R}_{2,2}} \end{bmatrix}$$

Now we need to show that $\mathbf{U}_2$ is computationally hidden. Let $\mathbf{E}_1 \in R^{m\times m}$, $\mathbf{E}_2 \in R^{m\times m}$, and $\mathbf{C}_3 \in R^{m\times m}$ be uniformly random matrices. By Theorem 4.2, we know that the following are indistinguishable from random:

$$\left(\widetilde{\mathbf{R}_{2,1}}, \mathbf{C}_1\widetilde{\mathbf{R}_{2,1}}\right) \quad \text{and} \quad \left(\widetilde{\mathbf{R}_{2,1}}, \mathbf{E}_1\right)$$

$$\left(\widetilde{\mathbf{R}_{2,2}}, \mathbf{C}_2\widetilde{\mathbf{R}_{2,2}}\right) \quad \text{and} \quad \left(\widetilde{\mathbf{R}_{2,2}}, \mathbf{E}_2\right)$$

$$\left(\widetilde{\mathbf{R}_{2,2}}, \mathbf{C}_3\widetilde{\mathbf{R}_{2,2}}\right) \quad \text{and} \quad \left(\widetilde{\mathbf{R}_{2,2}}, \mathbf{E}_3\right)$$

This allows us to reduce the previous matrix we were examining to

$$\begin{bmatrix} 0 & \mathbf{E}_2 - \mathbf{V}_1\mathbf{E}_3 \\ \mathbf{E}_1\mathbf{U}_2 + \mathbf{E}_3\mathbf{V}_2 & \mathbf{E}_1 + \mathbf{E}_3 \end{bmatrix}$$

Note that $\mathbf{E}_2 - \mathbf{V}_1\mathbf{E}_3$ is distributed randomly given the rest of the terms since $\mathbf{E}_2$ is random and is not present in any of the other terms. In order to finish the lemma, we need to show that the following two tuples are indistinguishable, where $\mathbf{Q} \in R^{m\times m}$ is a random matrix:

$$(\mathbf{U}_2, \mathbf{V}_2, \mathbf{E}_1\mathbf{U}_2 + \mathbf{E}_3\mathbf{V}_2, \mathbf{E}_1 + \mathbf{E}_3) \quad \text{and} \quad (\mathbf{U}_2, \mathbf{V}_2, \mathbf{E}_1\mathbf{Q} + \mathbf{E}_3\mathbf{V}_2, \mathbf{E}_1 + \mathbf{E}_3)$$

Observe that even if $\mathbf{E}_1$ is known to the adversary, the latter tuple is random since $(\mathbf{E}_1, \mathbf{E}_1\mathbf{Q})$ is indistinguishable from random by Theorem 4.2. It is enough to show that the tuple

$$\mathbf{U}_2, \mathbf{V}_2, \mathbf{E}_1\mathbf{U}_2 + \mathbf{E}_3\mathbf{V}_2, \mathbf{E}_1 + \mathbf{E}_3$$

is indistinguishable from random. We can write $\mathbf{F} = \mathbf{E}_1 + \mathbf{E}_3$ and then rewrite our tuple as

$$\mathbf{U}_2, \mathbf{V}_2, \mathbf{E}_1(\mathbf{U}_2 - \mathbf{V}_2) + \mathbf{F}\mathbf{V}_2, \mathbf{F}$$

Since we have eliminated $\mathbf{E}_3$ from the above tuple, we may assume that $\mathbf{F}$ is uniformly random and hidden from an adversary. Let $\mathbf{G} \in R^{m\times m}$ be a uniformly random matrix. By Theorem 4.2, we know that the following are indistinguishable from random:

$$(\mathbf{V}_2, \mathbf{F}\mathbf{V}_2) \quad \text{and} \quad (\mathbf{V}_2, \mathbf{G})$$

Then our tuple is indistinguishable from

$$\mathbf{U}_2, \mathbf{V}_2, \mathbf{E}_1(\mathbf{U}_2 - \mathbf{V}_2) + \mathbf{G}, \mathbf{F}$$

which is indistinguishable from random, completing the proof. □

11 Slotted RKHwPRFs

In this section, we define a weaker flavor of RKHwPRF called a "slotted" RKHwPRF. We begin by defining a plain slotted weak PRF with no algebraic structure.

Definition 11.1. (Slotted weak PRF.) An N-slotted weak PRF family is a set of weak PRF families $\{F_{i,j} : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}_{i,j}\}_{i,j\in[N],i\leq j}$ that share the same key space and input space, but may have different output spaces.

Each unit interval $[i, i]$ for $i \in [N]$ is referred to as a "slot". Note that for a unit interval $[i, i]$, we simply use the notations F_i and $\mathcal{Y}_i$ instead of $F_{i,i}$ and $\mathcal{Y}_{i,i}$.

We now define variants of these primitive with algebraic structure.

Definition 11.2. (Slotted RKHwPRF.) An N-slotted weak PRF family of the form $\{F_{i,j} : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}_{i,j}\}_{i,j \in [N], i \leq j}$ is an N-slotted RKHwPRF family if it satisfies the following properties:

- $(\mathcal{K}, \oplus, \otimes)$ is an efficiently samplable (finite) ring with efficiently computable ring operations.
- For each $i, j \in [N]$ such that $i \leq j$, we have the following:
  - $(\mathcal{Y}_{i,j}, \boxplus)$ is an efficiently samplable (finite) group with efficiently computable group operations.
  - For any $x \in \mathcal{X}$ the function $F_{i,j}(\cdot, x) : \mathcal{K} \to \mathcal{Y}_{i,j}$ is a group homomorphism, i.e., for any $x \in \mathcal{X}$ and $k, k' \in \mathcal{K}$ we have
$$F(k \oplus k', x) = F(k, x) \boxplus F(k', x).$$
- For each $i, j, \ell \in [N]$ such that $i \leq j < \ell$, there exists an efficiently computable function $\phi_{i,j,\ell} : \mathcal{Y}_{i,j} \times \mathcal{Y}_{j+1,\ell} \to \mathcal{Y}_{i,\ell}$ such that for any $x \in \mathcal{X}$ and $k, k' \in \mathcal{K}$, we have
$$F_{i,\ell}(k \otimes k', x) = \phi_{i,j,\ell}(F_{i,j}(k, x), F_{j+1,\ell}(k', x)).$$

11.1 Two-slotted-RKHwPRFs from Bilinear SXDH

Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be an asymmetric efficiently computable non-degenerate bilinear map with "source groups" $\mathbb{G}_1$ and $\mathbb{G}_2$, and "target group" $\mathbb{G}_T$ with generator g_T, where each group has order q (assumed prime). Also, let $\mathcal{X}$ be a set of size q such that there exist efficiently computable "encoding functions":

$$H_1 : \mathcal{X} \to \mathbb{G}_1, \quad H_2 : \mathcal{X} \to \mathbb{G}_2, \quad H_T : \mathcal{X} \to \mathbb{G}_T,$$

such that for any $x \in \mathcal{X}$, we have

$$H_T(x) = e(H_1(x), H_2(x)).$$

Now, define the functions $F_1 : \mathbb{Z}_q \times \mathcal{X} \to \mathbb{G}_1$ and $F_2 : \mathbb{Z}_q \times \mathcal{X} \to \mathbb{G}_2$ as:

$$F_1(k, x) = H_1(x)^k \quad , \quad F_2(k, x) = H_2(x)^k.$$

Similarly, define the function $F_{1,2} : \mathbb{Z}_q \times \mathcal{X} \to \mathbb{G}_T$ as:

$$F_{1,2}(k, x) = H_T(x)^k.$$

Assuming that bilinear SXDH holds (i.e., that the DDH assumption holds over each individual source group $\mathbb{G}_1$ and $\mathbb{G}_2$, and hence, over the target group $\mathbb{G}_T$), and assuming that the "encoding functions" H_1 and H_2 are *permutations*, we can state the following:

- F_1, F_2 and $F_{1,2}$ are weak PRFs.
- F_1 and F_2 are homomorphic with respect to addition. More concretely, for any $k_1, k_2 \in \mathbb{Z}_q$ and for any $x \in \mathcal{X}$, we have:
$$F_1(k_1 + k_2, x) = F_1(k_1, x) \cdot F_1(k_2, x) \quad , \quad F_2(k_1 + k_2, x) = F_2(k_1, x) \cdot F_2(k_2, x).$$
  Similarly, $F_{1,2}$ is also homomorphic with respect to addition. More concretely, for any $k_1, k_2 \in \mathbb{Z}_q$ and for any $x \in \mathcal{X}$, we have:
$$F_{1,2}(k_1 + k_2, x) = F_{1,2}(k_1, x) \cdot F_{1,2}(k_2, x).$$
  To see this observe that:
$$\begin{aligned} F_1(k_1 + k_2, x) &= H_1(x)^{k_1+k_2} \\ &= H_1(x)^{k_1} \cdot H_1(x)^{k_2} \\ &= F_1(k_1, x) \cdot F_1(k_2, x). \end{aligned}$$
  The addtitive homomorphisms of F_2 and $F_{1,2}$ follow similarly.

This gives us group key-homomorphic weak PRFs. However, defining ring-homomorphism is tricky, since none of $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ are necessarily equipped with efficiently computable multiplication operations (in fact, for some of these groups, such an operation may not even be properly defined).

However, note that we can use the bilinear map e to "simulate" a single multiplication operation as follows: for any $k_1, k_2 \in \mathbb{Z}_q$ and for any $x \in \mathcal{X}$, define the following operation:

$$\phi(F_1(k_1, x), F_2(k_2, x)) := e(F_1(k_1, x), F_2(k_2, x)).$$

It is easy to see the following:

$$\begin{aligned}
\phi(F_1(k_1, x), F_2(k_2, x)) &= e(F_1(k_1, x), F_2(k_2, x)) \\
&= e\left(H_1(x)^{k_1}, H_2(x)^{k_2}\right) \\
&= e\left(H_1(x), H_2(x)\right)^{k_1 \cdot k_2} \\
&= H_T(x)^{k_1 \cdot k_2} \\
&= F_{1,2}(k_1 \cdot k_2, x).
\end{aligned}$$

It is now easy to see that the ensemble $\{F_1, F_2, F_T\}$ is a two-slotted RKHwPRF family by construction.

11.2 Multi-slotted-RKHwPRFs from SXDH

We now generalize the aforementioned framework to generic asymmetric multilinear map of arbitrary degree n equipped with the SXDH assumption. We adopt the same definition of asymmetric multilinear maps presented in [GGH13a]. According to this definition, in asymmetric multilinear maps, the groups are indexed by integer vectors. Formally, a standard asymmetric multilinear map consists of the following .

- **Setup**$(1^\lambda, 1^N, \mathbf{n})$: Takes as input a vector $\mathbf{n} \in \mathbb{Z}^N$. Sets up an N-linear map by outputting a succinct description of groups $(\mathbb{G}_\mathbf{1}, \mathbb{G}_\mathbf{2}, \cdots, \mathbb{G}_\mathbf{n})$ of prime order q (where q is a λ bit prime), along with the respective generators $g_\mathbf{v} \in \mathbb{G}_\mathbf{v}$ for $\mathbf{1} \leq \mathbf{v} \leq \mathbf{n}$ (comparison of vectors is defined component-wise). Further, let $\mathbf{e}_i$ be the i-th *standard* basis vector (with 1 at position i and 0 at each other position). In standard notation, $\mathbb{G}_{\mathbf{e}_i}$ is the ith source group, $\mathbb{G}_\mathbf{n}$ is the target group, and the rest are the intermediate groups.

- $e_{\mathbf{v_1},\mathbf{v_2}}(h_1, h_2)$: Takes as input $h_1 \in \mathbb{G}_{\mathbf{v_1}}$ and $h_2 \in \mathbb{G}_{\mathbf{v_2}}$ subject to the restriction that $\mathbf{v_1} + \mathbf{v_2} <= \mathbf{n}$, and outputs $h_3 \in \mathbb{G}_{\mathbf{v_1}+\mathbf{v_2}}$ such that

  $$(h_1 = g_{\mathbf{v_1}}^a, h_2 = g_{\mathbf{v_2}}^b) \Rightarrow h_3 = g_{\mathbf{v_1}+\mathbf{v_2}}^{ab}$$

  For simplicity, we omit the subscripts and simply refer to this multilinear map as e, which may be generalized to multiple inputs as:

  $$e(h_1, \ldots, h_\ell) = e(h_1, e(h_2, \ldots, h_\ell)).$$

Now, analogous to the bilinear setting, let $\mathcal{X}$ be a set of size q and assume that there exists a family of encoding functions $\{H_{\mathbf{v}_i} : \mathcal{X} \to \mathbb{G}_{\mathbf{v}_i}\}_{\mathbf{v}_i <= \mathbf{n}}$ such that for any $\mathbf{v}_i, \mathbf{v}_j$ such that $\mathbf{v}_i + \mathbf{v}_j <= \mathbf{n}$, the following relation holds:

$$H_{\mathbf{v}_i+\mathbf{v}_j}(x) = e(H_{\mathbf{v}_i}(x), H_{\mathbf{v}_j}(x)).$$

Note that this can be achieved via the following steps:

- For the i-th standard basis vector $\mathbf{e}_i$, choose a *basis encoding function*:

  $$H_{\mathbf{e}_i} : \mathcal{X} \to \mathbb{G}_{\mathbf{e}_i}.$$

- For any vector $\mathbf{v} <= \mathbf{n}$ such that $\mathbf{v} = \sum_{i=1}^{N} \alpha_i \cdot \mathbf{e}_i$, define
$$H_{\mathbf{v}} := e(g_{\mathbf{e}_1}^{\alpha_1}, \ldots, g_{\mathbf{e}_N}^{\alpha_N}).$$

Next, for any any vector $\mathbf{v} <= \mathbf{n}$, define the function $F_{\mathbf{v}} : \mathbb{Z}_q \times \mathcal{X} \to \mathbb{G}_{\mathbf{v}}$ as:
$$F_{\mathbf{v}}(k, x) = H_{\mathbf{v}}(x)^k.$$

Assuming that SXDH holds (i.e., that the DDH assumption holds over each group $\mathbb{G}_{\mathbf{v}}$) and assuming that each *basis encoding function* $H_{\mathbf{e}_i}$ is a *permutation*, we can state the following:

- Each function $F_{\mathbf{v}}$ is a weak PRF.
- Each function $F_{\mathbf{v}}$ is homomorphic with respect to addition. More concretely, for any $k_1, k_2 \in \mathbb{Z}_q$ and for any $x \in \mathcal{X}$, we have:
$$F_{\mathbf{v}}(k_1 + k_2, x) = F_{\mathbf{v}}(k_1, x) \cdot F_{\mathbf{v}}(k_2, x).$$

This gives us group key-homomorphic weak PRFs. Again, defining ring-homomorphism is tricky, since none of the groups output by the setup algorithm are necessarily equipped with efficiently computable multiplication operations. However, yet again, we can use the multilinear map e to "simulate" multiplication operations as follows: for any $k_1, k_2 \in \mathbb{Z}_q$, for any $x \in \mathcal{X}$, and for any any $\mathbf{v}_i, \mathbf{v}_j$ such that $\mathbf{v}_i + \mathbf{v}_j <= \mathbf{n}$, define the following operation:
$$\phi(F_{\mathbf{v}_i}(k_1, x), F_{\mathbf{v}_j}(k_2, x)) := e(F_{\mathbf{v}_i}(k_1, x), F_{\mathbf{v}_j}(k_2, x)).$$

It is easy to see the following:
$$\begin{aligned} \phi(F_{\mathbf{v}_i}(k_1, x), F_{\mathbf{v}_j}(k_2, x)) &= e(F_{\mathbf{v}_i}(k_1, x), F_{\mathbf{v}_j}(k_2, x)) \\ &= e\left(H_{\mathbf{v}_i}(x)^{k_1}, H_{\mathbf{v}_j}(x)^{k_2}\right) \\ &= e\left(H_{\mathbf{v}_i}(x), H_{\mathbf{v}_j}(x)\right)^{k_1 \cdot k_2} \\ &= H_{\mathbf{v}_i + \mathbf{v}_j}(x)^{k_1 \cdot k_2} \\ &= F_{\mathbf{v}_i + \mathbf{v}_j}(k_1 \cdot k_2, x). \end{aligned}$$

Finally, we show how to concretely construct an N-slotted RKHwPRF family based on the aforementioned discussion:

- For each $i \in [N]$, define $\widetilde{F}_i$ (equivalently, $\widetilde{F}_{i,i}$) as $\widetilde{F}_i := F_{\mathbf{e_i}}$.
- For each $i, j \in [N]$ such that $i < j$, define $\widetilde{F}_{i,j} := F_{\mathbf{v}_{i,j}}$ where $\mathbf{v}_{i,j} = \sum_{\ell=i}^{j} \mathbf{e}_\ell$.

It is easy to see that the ensemble $\{\widetilde{F}_{i,j}\}_{i,j \in [N], i \leq j}$ is an N-slotted RKHwPRF family by construction.

11.3 Constructions from Slotted RKHwPRFs

We note that the key difference between classic and slotted RKHwPRFs is that the latter primitive is restricted in terms of "multiplication." Namely the multiplication operation is only defined between wPRF evaluations from "adjacent" slots, and the maximum multiplicative depth is bounded by the number of slots, which is a parameter of the slotted RKHwPRF family. However, these restrictions do not hinder our constructions from classic RKHwPRFs from a functional point of view. We elaborate more on this below.

Multiplying in Order. Our NIKE and iO constructions only require elements from the output ring of a classic RKHwPRF (or more generally, an RHS) to be multiplied in a pre-determined order. For example, in our NIKE construction, evaluating the final secret key requires the public/secret matrices of ring elements from various parties to be multiplied in a specific pre-determined order (informally, in the order in which the parties are indexed).

The restriced order of multiplications is even more natural in the case of our iO construction. From a functional point of view, evaluating a branching program using our iO construction requires the evaluator to multiply matrices of ring elements in the order of the level/variable index that they correspond to (depending on whether the matrix is a "program-carrying" matrix or an "enforcer" matrix). In addition, from a security point of view, we actually want to restrict an adversary from being able to multiply these matrices out of order, in order to prevent mix-and-match/input inconsistency attacks. This is the key rationale behind our design of "guard matrices" which, at a high level, enforce the restriction that only matrices (or in some cases submatrices) corresponding to adjacent levels/consecutive input variables can be meaningfully multiplied.

So, the lack of ability to multiply elements "out of order" certainly does not hurt us and may even help us in terms of practical security.

Pre-Determined Multiplicative Depth. Both our NIKE and iO constructions have a pre-fixed multiplicative depth. In the case of the NIKE construction, the multiplicative depth of the key derivation circuit is $(N-1)$, where N is the number of parties participating in the protocol. In the case of the iO construction, the multiplicative depth of the circuit evaluating an obfuscated program is bounded by $c(L+N)$, where L is the depth of the permutation branching program corresponding to the circuit to be obfuscated, N is the number of input variables and c is a fixed parameter of the construction. In particular, for our iO construction, the multiplicative depth of the evaluation circuit is independent of the actual circuit being obfuscated.

Sampling from Special Distributions. Our iO construction from classic RKHwPRF requires the ability to efficiently sample matrices of ring elements from certain special distributions. For example, the obfuscation algorithm samples guard matrices that are uniform subject to specific constraints; in particular, the products of certain guard submatrices are required to be zero. In the slotted RKHwPRF setting, the output spaces of weak PRFs are not rings and hence do not necessarily support efficient multiplication. This means we cannot sample such specially distributed matrices directly from the output spaces.

However, we can easily get around this issue by sampling from the key space instead. Recall that the key space for a slotted RKHwPRF is a ring, supports efficient multiplications, and is shared across all unit interval-levels. Hence, we can sample from the key space of the slotted RKHwPRF in exactly the same way as from the output space of a classic RKHwPRF. Next, we translate these sampled keys to the respective output spaces by evaluating the weak PRF at each interval-level on the same uniform input x. By the ring-homomorphic properties of the slotted RKHwPRF, the corresponding weak PRF outputs satisfy the desired distributions.

Overview of Construction Strategies. Based on the aforementioned observations, our constructions work from slotted RKHwPRFs in the same way as they work from classic RKHwPRFs. For example, intuitively, the NIKE construction for N parties can be instantiated from a $(2N-1)$-slotted RKHwPRF, where the N "odd-indexed" slots are used by each party to encode their secrets, and the "odd-indexed" are used for the public parameter matrices. In some more detail, the secret $\mathbf{S}_i$ for each party P_i would be a matrix of elements that belong to the unit interval $[2i-1, 2i-1]$, and each public parameter matrix $\mathbf{R}_i$ would be a matrix of elements that belong to the unit interval $[2i, 2i]$. The final secret key would be a matrix of elements at the top-level interval $[1, 2N-1]$.

Similarly, the iO construction for permutation branching programs with depth L and N variables can be instantiated from a $(c(L+N))$-slotted RKHwPRF, where c is a fixed parameter of the construction. Separate unit interval-slots would be designated for each program-carrying and enforcing matrix, as well as for the guard matrices, such that matrices in these slots can be sampled from appropriate distributions and multiplied in order by the obfuscation and evaluation algorithms.

Finally, observe that just like a classic RKHwPRF, a slotted RKHwPRF is inherently equipped with a zero test; evaluating a PRF at any interval-level $[i, j]$ with the zero key is guaranteed to produce the zero lement corresponding to the interval-level $[i, j]$. Hence, the zero test for our iO construction from a slotted RKHwPRF would also work in exactly the same way as the zero test for our iO construction from a classic RKHwPRF.

Arguing Security. Finally, the security arguments for our constructions from classic RKHwPRFs can be naturally adopted to work for their counteparts built from slotted RKHwPRFs. We note that the key space for a slotted RKHwPRF family is a ring (same as in a classic RKHwPRF), so the same (leftover hash lemma+weak pseudorandmoness) based arguments for "repeated" subset sums and general linear sums also apply to the slotted RKHwPRF setting. At a high level, an adversary against a slotted RKHwPRF has strictly lesser computational capabilities than an adversary against a classic RKHwPRF (due to the restrictions on the multiplication operations), and hence all relevant hardness assumptions naturally translate from the classic to the slotted setting.

12 Slotted Partial RKHwPRFs

In this section, we define another flavor of slotted RKHwPRF called a slotted partial RKHwPRF.

Definition 12.1. (Slotted Partial RKHwPRF.) An N-slotted partial RKHwPRF family is a collection of weak PRF families $\{F_{i,j} : (\mathcal{K} \times \mathcal{R}_{i,j}) \times \mathcal{X} \to \mathcal{Y}_{i,j}\}_{i,j \in [N], i \leq j}$ that satisfies the following properties:

- $(\mathcal{K}, \oplus, \otimes)$ is an efficiently samplable (finite) ring with efficiently computable ring operations.
- For each $i, j \in [N]$ such that $i \leq j$, we have the following:
  - $(\mathcal{Y}_{i,j}, \boxplus)$ is an efficiently samplable (finite) group with efficiently computable group operations.
  - There exists an efficiently computable function $\psi_{i,j} : \mathcal{R}_{i,j} \times \mathcal{R}_{i,j} \to \mathcal{R}_{i,j}$ such that for any $x \in \mathcal{X}$, any $k, k' \in \mathcal{K}$ and any $r, r' \in \mathcal{R}_{i,j}$, we have
  $$F((k \oplus k', r''), x) = F((k, r), x) \boxplus F((k', r'), x),$$
  where $r'' = \psi_{i,j}(r, r')$.
- For each $i, j, \ell \in [N]$ such that $i \leq j < \ell$, there exists:
  1. an efficiently computable function $\phi_{i,j,\ell} : \mathcal{Y}_{i,j} \times \mathcal{Y}_{j+1,\ell} \to \mathcal{Y}_{i,\ell}$, and
  2. an efficiently computable function $\psi_{i,j,\ell} : \mathcal{R}_{i,j} \times \mathcal{R}_{j+1,\ell} \to \mathcal{R}_{i,\ell}$,

  such that for any $x \in \mathcal{X}$, any $k, k' \in \mathcal{K}$, any $r \in \mathcal{R}_{i,j}$ and any $r' \in \mathcal{R}_{j+1,\ell}$ we have
  $$F_{i,\ell}((k \otimes k', r''), x) = \phi_{i,j,\ell}(F_{i,j}((k, r), x), F_{j+1,\ell}((k', r'), x)),$$
  where $r'' = \psi_{i,j,\ell}(r, r')$.
- There exists an efficiently computable function $\mathsf{ZeroTest}$ such that for $x \in \mathcal{X}$ and any $r \in \mathcal{R}_{i,j}$, the following holds with overwhelmingly large probability:
  $$\mathsf{ZeroTest}(F_{1,N}((k, r), x)) = 1 \text{ if and only if } k = 0_{\mathcal{K}},$$
  where $F_{1,N}$ is referred to as the *top-level PRF*.

Remark 12.2. Note that the zero test does not appear as an explicit requirement in the definitions of classical/slotted RKHwPRF presented earlier. This is because these primitives are inherently equipped with a zero test by virtue of their *exact* ring-homomorphism properties. More concretely, evaluating a classical/slotted RKHwPRF on any input using the key $k = 0_{\mathcal{K}}$ trivially results in a zero output, which simply serves as the zero test. However, this is not guaranteed when the homomorphism is only partial, in which case the zero test needs to be listed as an explicit requirement.

12.1 Partial Slotted RKHwPRF from the MZ18 MMap

Ma and Zhandry [MZ18] proposed a candidate polynomial-degree multilinear map scheme (referred to as the MZ18 MMap henceforth), that builds on top of the candidate polynomial-degree multilinear map scheme of Coron *et al.* [CLT13] (referred to as the CLT13 MMap henceforth). The MZ18 MMap construction is provably secure in the weak multilinear map model under the branching program unannihilatability assumption of Garg *et al.* [GMM$^+$16]. In particular, it provably subverts many of the zeroizing attacks [CLLT17] proposed against the original CLT13 MMap scheme.

In this section, we show that the MZ18 MMap implies a slotted partial RKHwPRF family. We begin with a description of the MZ18 MMap construction. We subsequently show an explicit construction of slotted partial RKHwPRF from the MZ18 MMap.

Overview of the MZ18 MMap. We provide an overview of the MZ18 MMap construction. The plaintext space for the construction is a ring $R = \mathbb{Z}_M$ (where M is not made public) with well-defined and efficiently computable addition and multiplication operations. The construction maps plaintext elements onto *encodings*, where each encoding is associated with a particular level. An MZ18 MMap of degree $N = \mathrm{poly}(\lambda)$ (where λ is the security parameter) supports a total of N^2 interval-levels of the form $[i, j]$ such that $i, j \in [N]$ and $i \leq j$. Each interval of the form $[i, i]$ for $i \in [n]$ is referred to as a singleton/unit interval-level.

At a high level, two kinds of operations are defined over the MZ18 MMap encodings - addition and multiplication. The addition operation is only defined between encodings that belong to the same interval-level $[i, j]$, while the multiplication operation is only defined over encodings at adjacent interval-levels, i.e., it is only possible to multiply encodings at levels $[i, j]$ and $[j + 1, k]$ such that $i, j, k \in [N]$ and $i \leq j < k$. The addition and multiplication operations define a partial ring homomorphism between the space of encodings and the ring of plaintext elements. Adding encodings of plaintext elements a and b produces an encoding of $(a + b)$. Similarly, multiplication encodings of a and b produces an encoding of $(a \cdot b)$.

Format of Encodings. Each encoding in the MZ18 MMap is a matrix, organized logically into NL columns. The columns are further partitioned into N groups (numbered 1 through N) consisting of L columns each. The columns in each group are interleaved; for each $i \in N$, the i^{th} group consists of the columns $i, (i+N), (i+2N), \ldots, (i+(L-1)N)$. The unit interval-level $[i, i]$ is essentially made up of the columns in the i^{th} group. Each MZ18 encoding at the unit interval-level $[i, i]$ consists of L matrices of CLT13 encodings, one in each of the columns corresponding to the column group i. More concretely, given a plaintext element $a \in \mathbb{Z}_m$, its overall meta-encoding at interval-level $[i, i]$ is given by the sequence of encodings $(\mathbf{A}_\ell^{(i)})_{\ell \in [L]}$ corresponding to the sequence of plaintext elements $(a, 0, \ldots, 0)$, where for each $\ell \in [L]$, $\mathbf{A}_\ell^{(i)}$ denotes the ℓ^{th} matrix in the encoding.

To construct $\mathbf{A}_\ell^{(i)}$, MZ18 uses a core diagonal matrix $\widetilde{\mathbf{A}}_\ell^{(i)}$ of the form

$$\widetilde{\mathbf{A}}_\ell^{(i)} = \begin{bmatrix} a_\ell &&&&&&&&& \\ & v_\ell &&&&&&&& \\ && w_\ell &&&&&&& \\ &&& \psi_1 \mathbf{I} &&&&&& \\ &&&& \ddots &&&&& \\ &&&&& \psi_{i-1}\mathbf{I} &&&& \\ &&&&&& \mathbf{E}_\ell^{(i)} &&& \\ &&&&&&& \psi_{i+1}\mathbf{I} && \\ &&&&&&&& \ddots & \\ &&&&&&&&& \psi_N \mathbf{I} \end{bmatrix}$$

where:

- $a_\ell \in \mathbb{Z}_M$ is the plaintext element being encoded.

- v_ℓ and w_ℓ are freshly sampled uniformly random elements from the plaintext space $\mathbb{Z}_M$ with the purpose of enforcing a requirement called *non-shortcutting* that is essential to the security proof for the MZ18 construction. We refer the reader to [MZ18] for more details.

- Each block matrix of the form $\psi_j\mathbf{I}$ for $j \in [N] \setminus \{i\}$ is a random multiple of the identity matrix that simply serves as a placeholder and is essentially unused.

- The block matrix $\mathbf{E}_\ell^{(i)}$ is an *enforcer matrix* with the purpose of preventing an adversary from arbitrarily mixing and matching the matrices from different encodings. We again refer the reader to [MZ18] for more details on how the enforcer matrix is constructed.

Kilian Randomization. A total of $(NL+1)$ Kilian randomization matrices are generated, indexed as $\mathbf{R}_0, \mathbf{R}_1, \ldots, \mathbf{R}_{NL}$. s. All encodings share the same Kilian matrices. Each $\widetilde{\mathbf{A}}_\ell^{(i)}$ matrix at meta-level i is left- and right- multiplied by the corresponding Kilian randomization matrices, giving the encoding

$$\widetilde{\mathbf{A}'}^{(i)} = \mathbf{R}_{N(\ell-1)+(i-1)}\widetilde{\mathbf{A}}_\ell^{(i)}\mathbf{R}_{N(\ell-1)+i}.$$

The Final Encoding. Finally, the MZ18 encoding $\mathbf{A}_\ell^{(i)}$ is generated as

$$\mathbf{A}_\ell^{(i)} = \mathsf{Enc\text{-}CLT}(\widetilde{\mathbf{A}'}^{(i)}, N(\ell-1)+i),$$

where the function $\mathsf{Enc\text{-}CLT}$ takes as input a matrix containing elements in $\mathbb{Z}_N$ and a level z, and outputs a matrix such that each entry of the output matrix is a CLT13 encoding of the corresponding entry in the input matrix at the singleton level-set $\{z\}$.

We avoid presenting the details of the CLT13 encoding algorithm; the reader may refer [CLT13, MZ18]. Instead, we highlight certain properties of the CLT13 encoding that are essential to the construction of slotted partial RKHwPRFs from the MZ18 MMap:

- There exists an efficiently computable addition operation over CLT13 encodings that belong to the same level-set. The addition operation between encodings defines a group that is homomorphic to the additive group over the plaintext space.

- There exists an efficiently computable multiplication operation over CLT13 encodings that belong to disjoint level-sets. The resulting product is an encoding of the product of the underlying plaintext elements, albeit at the level-set which is the union of the input level-sets.

We will subsequently see how these partial ring-homomorphic properties in the original CLT13 encoding are inherited by the MZ18 MMap encoding. We also note that the interval-levels supported by the MZ18 encoding are essentially a restriction of the level-sets supported by the CLT13 encoding, and this restriction plays a key role in preventing the zeroizing attacks plaguing the original CLT13 MMap construction.

Operations over Encodings. We now describe the addition and multiplication operations over the MZ18 encodings. Recall that given a plaintext element $a \in \mathbb{Z}_m$, its encoding is given by the sequence of encodings $(\mathbf{A}_\ell^{(i)})_{\ell\in[L]}$ corresponding to the sequence of plaintext elements $(a, 0, \ldots, 0)$. Before describing the operations, we assume that the following encodings are published at setup:

- Encodings of $\{1, 2, 4, \ldots, 2^{\rho-1}\}$ at every interval-level $[i, j]$ such that $i, j \in [N]$ and $i \leq j$, where τ' is a parameter that also depends on the plaintext ring $\mathbb{Z}_M$.

- τ'-many uniformly random encodings of 0 at every interval-level $[i, j]$ such that $i, j \in [N]$ and $i \leq j$, where τ' is a parameter that also depends on the plaintext ring $\mathbb{Z}_M$.

Addition. To add two encodings at the same unit interval-level $[i, i]$, which are essentially two sequences of matrices with entries in $\mathbb{Z}_M$, one lines up the sequences of matrices and adds the corresponding matrices component-wise. The resulting sequence of matrices is taken as the encoding of the sum. Intuitively, this works because:

1. All encodings at the same level use the same pair of Kilian randomizers; hence adding these matrices also adds the sequence of underlying plaintexts.

2. The enforcing matrices corresponding to the same level are generated in a manner that preserves their structure across addition over $\mathbb{Z}_M$.

More concretely, if the input encodings have plaintext sequences $(a_1, 0, \ldots, 0)$ and $(a_2, 0, \ldots, 0)$, the result of addition has plaintext sequence $(a_1 + a_2, 0, \ldots, 0)$.

Multiplication. To multiply two encodings at interval-levels $[i, j]$ and $[j+1, j']$ for $i, j, j' \in [N]$ such that $i \leq j < j'$, one proceeds as follows: for each $\ell \in [L]$, it multiplies the ℓ^{th} matrix from the first encoding with the ℓ^{th} matrix of the second encoding, re-randomizes it by adding the resulting matrix to a random encoding of 0 at the level $[i, j']$, and sets the resulting matrix to be the ℓ^{th} matrix of the output encoding. Again, intuitively, this works because:

1. For each $\ell \in [L]$, the ℓ^{th} matrices in the meta-encodings at the interval-levels $[i, j]$ and $[j + 1, j']$ use the Kilian randomizer pairs $(\mathbf{R}_{N(\ell-1)+(i-1)}, \mathbf{R}_{N(\ell-1)+j})$ and $(\mathbf{R}_{N(\ell-1)+j}, \mathbf{R}_{N(\ell-1)+(j'-1)})$, and hence multiplying these results in a matrix with the the Kilian randomizer pair $(\mathbf{R}_{N(\ell-1)+(i-1)}, \mathbf{R}_{N(\ell-1)+j'})$, as desired for the interval-level $[i, j']$.

2. Multiplying the matrices also multiplies (component-wise) the sequence of underlying plaintexts. This follows from the structure of the meta-encoding matrices.

3. For each $\ell \in [L]$, the pair of enforcing matrices $(\mathbf{E}_\ell^{(i,j)}, \mathbf{E}_\ell^{(j+1,j')})$ corresponding to the interval-levels $[i, j]$ and $[j + 1, j']$ is structured in a manner such that their product matrix $\mathbf{E}^{(i,j')} = \mathbf{E}_\ell^{(i,j)}\mathbf{E}_\ell^{(j+1,j')}$ is structured as an enforcing matrix corresponding to the interval-level $[i, j']$ should be structured.

More concretely, if the input encodings have plaintext sequences $(a_1, 0, \ldots, 0)$ and $(a_2, 0, \ldots, 0)$, the result of multiplication has plaintext sequence $(a_1 \cdot a_2, 0, \ldots, 0)$.

Generating Encodings of Arbitrary Plaintexts. We now describe how to efficiently generate MZ18 encodings corresponding to arbitrary plaintexts. Observe that we published at setup encodings of $\{1, 2, 4, \ldots, 2^{\rho-1}\}$ at each unit interval-level $[i, i]$ for $i \in [n]$, where ρ is a parameter that depends on the plaintext ring $\mathbb{Z}_M$. To encode a plaintext $a \in \mathbb{Z}_{2^\rho}$ at a unit interval-level $[i, i]$, one can then write the plaintext in base 2, and then sum the appropriate public encodings of powers of 2. In [MZ18], the authors propose setting $\rho = M \times 2^\lambda$ for a security parameter λ so that a random $a \in \mathbb{Z}_{2^\rho}$ yields a plaintext element $a' = a \mod M$ that is statistically close to random over the plaintext ring $\mathbb{Z}_M$.

Re-randomizing Encodings. The MZ18 MMap construction also supports re-randomization. Recall that we published (at setup) τ'-many uniformly random encodings of 0 at every interval-level $[i, j]$ such that $i, j \in [N]$ and $i \leq j$. In order to re-randomize any encoding at level $[i, j]$, add to the encoding a subset-sum of the public encodings of 0 available at this level.

Top-Level Zero Test. For the topmost interval-level $[1, N]$, the MZ18 construction publishes (at setup) a special pre-zero-test encoding that will have most of the structure of a valid top level encoding, except that it will not correctly encode an actual plaintext element. Instead, it encodes the plaintext-sequence will be $(0, 1, 1, \ldots, 1)$, which differs from a normal encoding where the plaintext-squences contain 0 at all but the first slot. The purpose of this encoding is to be added to any top level encoding we seek to zero test. In other words, suppose that we have a top-level encoding of zero, i.e., a top-level encoding of the plaintext-sequence $(0, 0, \ldots, 0)$. Then adding the pre-zero-test encoding converts it into a top-level encoding for the plaintext-sequence $(0, 1, 1, \ldots, 1)$. On the other hand, adding the pre-zero-test encoding to a non-zero encoding, i.e., a top-level encoding of the plaintext-sequence $(a, 0, \ldots, 0)$ for $a \neq 0$, converts it into a top-level encoding for the plaintext-sequence $(a, 1, 1, \ldots, 1)$.

The next step is to design a test that checks for the plaintext sequence $(0, 1, 1, \ldots, 1)$. To achieve this, the MZ18 construction also publishes (at setup) a pair of *bookend encodings*, which are CLT13 encodings of specially structured vectors $\mathbf{s}$ and $\mathbf{t}$ such that

$$\mathbf{s} = \begin{bmatrix} 1 & 1 & 0 & F_1 & \ldots & F_N \end{bmatrix} \mathbf{R}_0 \quad , \quad \mathbf{t} = \mathbf{R}_{NL}^{-1} \begin{bmatrix} 1 \\ 0 \\ 1 \\ G_1 \\ \vdots \\ G_N \end{bmatrix},$$

where the F and G vectors are structured to interact with the enforcing matrices in a manner that ensures that multiplying a top-level encoding for the plaintext-sequence $(a, 1, 1, \ldots, 1)$ with the bookend encodings results in a CLT13 encoding of the plaintext a. At this point, one can directly invoke the zero test procedure for the CLT13 MMap.

Formal Definition. Based on the description of the MZ18 MMap above, we formally represent the MZ18 MMap construction as an ensemble of the following poly-time algorithms:

- $\mathsf{Setup\text{-}MZ}(1^\lambda, N)$: Takes as input a security parameter λ and the number of levels L and outputs a public parameter pp, which contains the following:
  1. Sufficiently many encodings of powers of 2 at all interval levels.
  2. Sufficiently many encodings of 0 and sufficiently many encodings of 1 at all interval-levels.
  3. The pre-zero test encoding and the bookend encodings for the zero test.

- $\mathsf{Enc\text{-}MZ}(\mathsf{pp}, a, (i, j); r)$: Takes as input the public parameter pp, a plaintext element $a \in \mathbb{Z}_M$, indices $i, j \in [N]$ such that $i \leq j$ and random coins r, and generates a sequence of randomized encodings $(\mathbf{A}_\ell^{(i,j)})_{\ell \in [L]}$ corresponding to the interval-level $[i, j]$.

- $\mathsf{ReRand\text{-}MZ}(\mathsf{pp}, (\mathbf{A}_\ell^{(i,j)})_{\ell \in [L]}, (i, j); r)$: Takes as input pp, a sequence of encodings $(\mathbf{A}_\ell^{(i,j)})_{\ell \in [L]}$ corresponding to the interval-level $[i, j]$ and random coins r, and generates $(\widehat{\mathbf{A}}_\ell^{(i,j)})_{\ell \in [L]}$, which is a re-randomization of the input encoding sequence at the same level $[i, j]$.

- $\mathsf{Add\text{-}MZ}(\mathsf{pp}, (\mathbf{A}_\ell^{(i,j)})_{\ell \in [L]}, (\mathbf{B}_\ell^{(i,j)})_{\ell \in [L]}, (i, j); r)$: Takes as input pp, two sequences of encodings $(\mathbf{A}_\ell^{(i,j)})_{\ell \in [L]}$ and $(\mathbf{B}_\ell^{(i,j)})_{\ell \in [L]}$ at the interval-level $[i, j]$, and random coins r, and outputs $(\mathbf{C}_\ell^{(i,j)})_{\ell \in [L]}$, which represents the sum of the input encodings at the same level $[i, j]$.

- $\mathsf{Mult\text{-}MZ}(\mathsf{pp}, (\mathbf{A}_\ell^{(i,j)})_{\ell \in [L]}, (\mathbf{B}_\ell^{(j+1,j')})_{\ell \in [L]}, (i, j, j'); r)$: Takes as input pp, two sequences of encodings $(\mathbf{A}_\ell^{(i,j)})_{\ell \in [L]}$ and $(\mathbf{B}_\ell^{(j+1,j')})_{\ell \in [L]}$ at the interval-levels $[i, j]$ and $[j + 1, j']$ respectively, and random coins r, and outputs $(\mathbf{C}_\ell^{(i,j')})_{\ell \in [L]}$ which represents the multiplication of the input encodings, albeit at the interval-level $[i, j']$.

- $\mathsf{ZeroTest\text{-}MZ}(\mathsf{pp}, (\mathbf{A}_\ell^{(1,N)})_{\ell \in [L]})$: Takes as input pp, a top-level encoding-sequence $(\mathbf{A}_\ell^{(1,N)})_{\ell \in [L]}$ and outputs either 0 or 1.

We avoid formal descriptions of these functions, which essentially follow from the informal description presented earlier. The readers may refer [MZ18] for the formal descriptions.

Choosing the Computational Assumption. We now focus on choosing the appropriate computational assumption over the MX18 MMap to build slotted partial RKHwPRFs. Note that the standard SXDH assumption in the context of the MZ18 MMap construction implies that the following assumption is true: letting $\mathsf{pp} = \mathsf{Setup\text{-}MZ}(1^\lambda, N)$, the following ensembles are computationally indistinguishable for any $i, j \in [N]$ auch that $i \leq j$ and any $T = \mathrm{poly}(\lambda)$:

$$(\mathsf{pp}, \{\mathsf{Enc\text{-}MZ}(\mathsf{pp}, a_t, (i,j); r_t), \mathsf{Enc\text{-}MZ}(\mathsf{pp}, k \cdot a_t, (i,j); r_t)\}_{t\in[T]})$$
$$\approx_c (\mathsf{pp}, \{\mathsf{Enc\text{-}MZ}(\mathsf{pp}, a_t, (i,j); r_t), \mathsf{Enc\text{-}MZ}(\mathsf{pp}, b_t, (i,j); r_t)\}_{t\in[T]}),$$

where $k, a_1, \dots, a_T, b_1, \dots, b_T$ are uniformly sampled plaintext elements in $\mathbb{Z}_M$, and $r_1, \dots, r_T$ are appropriately distributed random coins for the MZ18 encoding algorithm.

However, since the MZ18 MMap uses "noisy" encodings, it does not support scalar multiplication with encodings. In other words, there is no efficient algorithm to compute an MZ18 encoding of $(k.a)$ at interval-level $[i, j]$ given an MZ18 encoding of a at interval-level $[i, j]$. As a result, this assumption is not immediately useful to construct weak PRF families.

We overcome this difficulty by using the following alternative computational assumption called "shifted"-SXDH: letting $\mathsf{pp} = \mathsf{Setup\text{-}MZ}(1^\lambda, N)$, the following ensembles are computationally indistinguishable for any index $i \in [N]$ and any $T = \mathrm{poly}(\lambda)$:

$$(\mathsf{pp}, \{\mathsf{Enc\text{-}MZ}(\mathsf{pp}, a_t, (i,j); r_t), \mathsf{Enc\text{-}MZ}(\mathsf{pp}, k \cdot a_t, (i,j+1); r_t)\}_{t\in[T]})$$
$$\approx_c (\mathsf{pp}, \{\mathsf{Enc\text{-}MZ}(\mathsf{pp}, a_t, (i,j); r_t), \mathsf{Enc\text{-}MZ}(\mathsf{pp}, b_t, (i,j+1); r_t)\}_{t\in[T]}),$$

where $k, a_1, \dots, a_T, b_1, \dots, b_T$ are uniformly sampled plaintext elements in $\mathbb{Z}_M$, and $r_1, \dots, r_T$ are appropriately distributed random coins for the MZ18 encoding algorithm.

Now observe the following: given an MZ18 encoding of a at interval-level $[i, j]$, we can construct an MZ18 encoding of $(k.a)$ at level $[i, j + 1]$ via the following steps:

1. **Step-1:** Create an encoding of k at interval-level $[j + 1, j + 1]$.

2. **Step-2:** Invoke the MZ18 multiplication algorithm on the encoding of a at interval-level $[i, j]$ and the encoding of k at interval-level $[j + 1, j + 1]$.

In other words, by resorting to the "shifted"-SXDH assumption, we can construct a weak PRF family where each function is efficiently computable, and the function output is "shifted" from the function input by a level. This allows us to overcome the limitations arising out of using the original SXDH assumption. However, the following question arises:

Is the "shifted"-SXDH assumption stronger than the original SXDH assumption?

The answer is *no*: the "shifted"-SXDH assumption is, in fact, *implied by* the SXDH assumption. To see this, observe that given an instance of the SXDH assumption, a PPT algorithm can easily simulate an instance of the "shifted"-SXDH assumption using the multiplication algorithm over the MZ18 encodings, such that the latter is a valid "shifted"-SXDH instance if and only if the former is a valid SXDH instance. In other words, we can still base the security of our RKHwPRF family on the SXDH assumption.

Constructing Slotted Weak PRFs. Equipped with the aforementioned computational assumption, we now show how to concretely construct a slotted weak PRF family from the MZ18 MMap. Let $\mathsf{pp} = \mathsf{Setup\text{-}MZ}(1^\lambda, 2N + 1)$. Let $\mathcal{X}$ be a set of size M and assume that there exists a family of (randomized) hash functions $\{H_{i,j}\}_{i,j\in[N], i\leq j}$ such that for any $x \in \mathcal{X}$, for any $i, j, j' \in [N]$ satisfying $i \leq j < j'$, there exists appropriately distributed random coins r such that the following relation holds:

$$H_{i,j'}(x) = \mathsf{Mult\text{-}MZ}(\mathsf{pp}, H_{i,j}(x), H_{j+1,j'}(x), r).$$

Note that this can be achieved via the following steps:

- For each unit interval-level $[i,i]$ such that $i \in [N]$, choose a (randomized) hash function $H_{i,i}$ that maps a uniformly random element $x \in \mathcal{X}$ to a level-$[i,i]$ encoding for a uniformly random plaintext element $a \in \mathbb{Z}_m$.

- For any $i, j, j' \in [N]$ satisfying $i \leq j < j'$, define the (randomized) hash function:

$$H_{i,j'}(x) = \mathsf{Mult\text{-}MZ}(H_{i,j}(x), H_{j+1,j'}(x), r),$$

  where r represents appropriately distributed random coins.

We now show how to construct the slotted partial RKHwPRF family. For any $i, j \in [N]$ satisfying $i \leq j$, define the function $F_{i,j} : (\mathbb{Z}_M \times \mathcal{R}_{i,j}) \times \mathcal{X} \to \mathcal{Y}_{i,j}$ as:

$$F_{i,j}((k,(r_0,r_1,r_2)),x) = \mathsf{Mult\text{-}MZ}(\mathsf{pp}, H_{2i,2j}(x;r_0), \mathsf{Enc\text{-}MZ}(\mathsf{pp}, k, 2j+1; r_1), r_2),$$

where r_0, r_1 and r_2 represent appropriately distributed random coins for the hashing, MZ18 encoding and MZ18 multiplication algorithms, respectively. [1] It is easy to see the following: under the assumption that the SXDH assumption holds (and hence the "shifted"-SXDH assumption holds), the function $F_{i,j}$ corresponding to each valid interval-level $[i,j]$ is a weak PRF.

Partial Ring Homomorphism. We now demonstrate that the aforementioned slotted weak PRF family is in fact a slotted weak RKHwPRF family. More concretely, we show this slotted weak PRF family is equipped with the following properties: (a) additive homomorphism within interval-levels (b) partial multiplicative homomorphism across "adjacent" interval levels, and (c) an efficient top-level zero test.

Additive Homomorphism. We define the following (randomized) addition operation:

$$F_{i,j}((k,r),x) \boxplus F_{i,j}((k',r'),x) := \mathsf{Add\text{-}MZ}(\mathsf{pp}, F_{i,j}((k,r),x), F_{i,j}((k',r'),x), r''),$$

where r'' represents appropriately distributed random coins for the MZ18 addition operation.

Observe that the output of $F_{i,j}$ is always an MZ18 encoding sequence at the interval-level $[2i, 2j+1]$. Hence, the aforementioned multiplication operation is well-defined. Finally, additive homomorphism follows from the additive homomorphism of the MZ18 encodings.

Partial Multiplicative Homomorphism. Next, we define the following (randomized) multiplication operation:

$$F_{i,j}((k,r),x) \boxtimes F_{j+1,j'}((k',r'),x) := \mathsf{Mult\text{-}MZ}(\mathsf{pp}, F_{i,j}((k,r),x), F_{j+1,j'}((k',r'),x), r''),$$

where r'' represents appropriately distributed random coins for the MZ18 multiplication operation.

Again, observe that the output of $F_{i,j}$ and $F_{j+1,j'}$ are MZ18 encoding sequences at the interval-levels $[2i, 2j+1]$ and $[2j+2, 2j'+1]$, respectively. Hence, the aforementioned multiplication operation is well-defined, and results in an encoding at the level $[2i, 2j'+1]$, as desired. Additionally, partial multiplicative homomorphism follows from the partial multiplicative homomorphism of the MZ18 encodings.

Top Level Zero-Test. We now describe the $\mathsf{ZeroTest}$ algorithm for the top-level PRF. Note that the output of our top-level PRF $F_{1,n}$ is an MZ18 encoding sequence at the interval-levels $[2, 2N+1]$. Hence, to zero-test, we use the following steps:

1. **Step-1:** Multiply the output of the PRF $F_{1,n}$ with a publicly available encoding of 1 at the interval-level $[1,1]$ (note that such an encoding is available as part of the public parameter pp) generated by the $\mathsf{Setup\text{-}MZ}$ algorithm.

2. **Step-2:** Invoke the MZ18 zero test algorithm $\mathsf{ZeroTest\text{-}MZ}$ on the resulting encoding.

Correctness of the aforementioned zero test procedure follows from the partial multiplicative homomorphism of the MZ18 encodings and the correcness of the MZ18 zero test algorithm.

[1] We implicitly assume that the MZ18 public parameter pp is part of the PRF family description.

Is Level-Doubling Necessary? Our construction of slotted partial RKHwPRFs requires an MZ18 MMap instantiation with approximately twice the number of levels. This arises from the need to prove weak pseudorandomness. Recall that in the weak pseudorandomness security game, the adversary is allowed to see uniformly randomly sampled input-output pairs. In case of the MZ18 MMap which has noisy encodings, it seems hard to construct a weak PRF family based on the SXDH assumption, where each function satisfies inputs and outputs are encodings at the same level, while also preserving partial ring homomorphism. Hence, we resorted to constructing a weak PRF family where the output encodings are "shifted" from the input encodings by a single level. This doubles the overall number of levels required for the construction.

However, it is possible to avoid this doubling requirement on the MZ18 MMap instantiation if we constructed a slightly weaker primitive, namely a slotted partial ring-homomorphic synthesizer (RHS), which is in fact sufficient for our target applications, namely NIKE and iO. In the synthesizer security game, the adversary is only allowed to see the outputs of the synthesizer on uniformly random input pairs. Since the inputs are not made public, they need not be encoded into separate levels; hence, an MZ18 MMap with the same number of levels as the slotted RHS family suffices. This gives better concrete efficiency from the point of view of applications. In other words, *level-doubling is not necessary* when using our techniques to build NIKE/iO from the MZ18 MMap; it more of a definitional requirement.

Nonetheless, we chose to present the slotted partial RKHwPRF construction since it is a natural weakening of the classical RKHwPRF primitive defined in the main body of the paper, as well as the slotted RKHwPRF primitive presented in the previous section. In particular, it serves as a natural adaptation of the slotted RKHwPRF primitive (which can be built from classical graded encodings) in the context of noisy candidate MMaps, such as the MZ18 MMap. We leave it as an interesting open question to build slotted partial RKHwPRFs from well-studied computational assumptions over MZ18 MMaps (and other candidate nosiy MMaps) while avoiding the level-doubling requirements.

12.2 Partial Slotted RKHwPRF from Symmetric MMaps

In this section, we show that how to design partial slotted RKHwPRFs from symmetric multilinear maps. Note that the SXDH assumption is trivially broken over symmetric multilinear maps; so we resort to using the more general matrix DDH family of assumptions. We note that multilinear maps we use here do not need to be fully symmetric for our construction to work. In fact, any multilinear map where the matrix-DDH assumption holds will work for this construction, so this construction could be used for asymmetric multilinear maps. However, since SXDH clearly cannot hold in asymmetric multilinear maps, we emphasize the application to symmetric multilinear maps below.

Let $e : \mathbb{G} \times \mathbb{G} \times \ldots \times \mathbb{G} \rightarrow \mathbb{G}_T$ be a symmetric efficiently computable non-degenerate N-linear map with "source group" $\mathbb{G}$ and "target group" $\mathbb{G}_T$, where each group has order q (assumed prime), and let $g \leftarrow \mathbb{G}$ be a publicly available uniformly sampled generator element for $\mathbb{G}$.

Matrix DDH Assumption. Suppose that $n, m = \mathrm{poly}(\lambda)$ are arbitrarily large, subject to the restriction that $n > m$. We recall the $\mathcal{U}_{n,m}$-matrix DDH (MDDH) Assumption assumption from [EHK$^+$13] and present some useful associated lemmas/corollaries.

Definition 12.3. ($\mathcal{U}_{n,m}$-MDDH Assumption.) The $\mathcal{U}_{n,m}$-MDDH assumption is said to hold over the group $\mathbb{G}$ of prime order q if letting $\mathbf{A} \leftarrow \mathbb{Z}_q^{n\times m}$, $\mathbf{k} \leftarrow \mathbb{Z}_q^m$ and $\mathbf{u} \leftarrow \mathbb{Z}_q^n$, we have

$$(g^{\mathbf{A}}, g^{\mathbf{Ak}}) \approx_c (g^{\mathbf{A}}, g^{\mathbf{u}}),$$

where $g \leftarrow \mathbb{G}$ is a uniformly sampled generator element.

The following corollary also follows via a simple hybrid argument.

Corollary 12.4. *Let $L = \mathrm{poly}(\lambda)$ and $n = Lm$. Assuming that the $\mathcal{U}_{n,m}$-MDDH assumption holds over the group $\mathbb{G}$, we have*

$$\left\{(g^{\mathbf{A}_\ell}, g^{\mathbf{A}_\ell \mathbf{K}})\right\}_{\ell\in[L]} \approx_c (\left\{g^{\mathbf{A}_\ell}, g^{\mathbf{U}_\ell})\right\}_{\ell\in[L]},$$

where $g \leftarrow \mathbb{G}$ is a uniformly sampled generator element, $\mathbf{K} \leftarrow \mathbb{Z}_q^{m\times m}$, and for each $\ell \in [L]$, we have $\mathbf{A}_\ell, \mathbf{U}_\ell \leftarrow \mathbb{Z}_q^{m\times m}$.

We also state and prove the following statistical indistinguishability lemma.

Lemma 12.5. *For any $N, L = \mathrm{poly}(\lambda)$, we have*

$$\begin{bmatrix} g^{\mathbf{A}_1 \mathbf{K}_{1,1} \mathbf{A}_2^{-1}} & \dots & g^{\mathbf{A}_N \mathbf{K}_{1,N} \mathbf{A}_{N+1}^{-1}} \\ \vdots & \ddots & \vdots \\ g^{\mathbf{A}_1 \mathbf{K}_{L,1} \mathbf{A}_2^{-1}} & \dots & g^{\mathbf{A}_N \mathbf{K}_{L,N} \mathbf{A}_{N+1}^{-1}} \end{bmatrix} \approx_s \begin{bmatrix} g_{1,1}^{\mathbf{U}_{1,1}} & \dots & g_{N,N}^{\mathbf{U}_{1,N}} \\ \vdots & \ddots & \vdots \\ g_{1,1}^{\mathbf{U}_{L,1}} & \dots & g_{N,N}^{\mathbf{U}_{L,N}} \end{bmatrix}$$

where $g \leftarrow \mathbb{G}$ is a uniformly sampled generator element, for each $n \in [N+1]$, we have $\mathbf{A}_i \leftarrow \mathbb{Z}_q^{m \times m}$ and for each $(\ell, i) \in [L] \times [N]$, we have $\mathbf{K}_{\ell,i}, \mathbf{U}_{\ell,i} \leftarrow \mathbb{Z}_q^{m \times m}$.

Proof. To see that this lemma is true, observe the following:

$$\begin{aligned} \begin{bmatrix} g^{\mathbf{A}_1 \mathbf{K}_{1,1} \mathbf{A}_2^{-1}} & \dots & g^{\mathbf{A}_{N-1} \mathbf{K}_{1,N-1} \mathbf{A}_N^{-1}} & g^{\mathbf{A}_N \mathbf{K}_{1,N} \mathbf{A}_{N+1}^{-1}} \\ \vdots & \ddots & \vdots & \vdots \\ g^{\mathbf{A}_1 \mathbf{K}_{L,1} \mathbf{A}_2^{-1}} & \dots & g^{\mathbf{A}_{N-1} \mathbf{K}_{L,N-1} \mathbf{A}_N^{-1}} & g^{\mathbf{A}_N \mathbf{K}_{L,N} \mathbf{A}_{N+1}^{-1}} \end{bmatrix} &\approx_s \begin{bmatrix} g^{\mathbf{A}_1 \mathbf{K}_{1,1} \mathbf{A}_2^{-1}} & \dots & g^{\mathbf{A}_{N-1} \mathbf{K}_{1,N-1} \mathbf{A}_N^{-1}} & g^{\mathbf{A}_N \mathbf{U}'_{1,N}} \\ \vdots & \ddots & \vdots & \\ g^{\mathbf{A}_1 \mathbf{K}_{L,1} \mathbf{A}_2^{-1}} & \dots & g^{\mathbf{A}_{N-1} \mathbf{K}_{L,N-1} \mathbf{A}_N^{-1}} & g^{\mathbf{A}_N \mathbf{U}'_{L,N}} \end{bmatrix} \\ &\approx_s \begin{bmatrix} g^{\mathbf{A}_1 \mathbf{K}_{1,1} \mathbf{A}_2^{-1}} & \dots & g^{\mathbf{A}_{N-1} \mathbf{K}_{1,N-1} \mathbf{A}_N^{-1}} & g^{\mathbf{U}_{1,N}} \\ \vdots & \ddots & \vdots & \vdots \\ g^{\mathbf{A}_1 \mathbf{K}_{L,1} \mathbf{A}_2^{-1}} & \dots & g^{\mathbf{A}_{N-1} \mathbf{K}_{L,N-1} \mathbf{A}_N^{-1}} & g^{\mathbf{U}_{L,N}} \end{bmatrix} \end{aligned}$$

where for each $\ell \in [L]$, we have $\mathbf{U}_{\ell,N}, \mathbf{U}'_{\ell,N} \leftarrow \mathbb{Z}_q^{m \times m}$.

The proof of Lemma 12.5 now follows from a simple hybrid argument, where in hybrid-i we apply the same set of indistinguishable transformations as illustrated above to column $(N - i + 1)$ of the original matrix. $\square$

We now describe how to build a slotted partial RKHwPRF family from a symmetric multilinear map such that the aforementioned MDDH assumption and the associated lemmas hold over it.

Notation. We first set up some notation. For each $i \in [N]$, we define a unit interval group: $\mathbb{G}_{i,i} := \mathbb{G}$ and a unit interval generator $g_{i,i} := g$. For each $i, j, \ell \in [N]$ such that $i \le j < \ell$, we define the map

$$e_{i,j,\ell} : \mathbb{G}_{i,j} \times \mathbb{G}_{j+1,\ell} \to \mathbb{G}_{i,\ell},$$

and the group element

$$g_{i,\ell} := e_{i,j,\ell}(g_{i,j}, g_{j+1,\ell}).$$

We also overload the notation e and use it instead of $e_{i,j,\ell}$ whenever the source and target groups are clear from the context. It is easy to see that the "top-level" group $\mathbb{G}_{1,N}$ is essentially the final group $\mathbb{G}_T$ and the group element $g_{1,N}$ is essentially a generator for $\mathbb{G}_T$.

The Slotted Function Family. Let $m = \mathrm{poly}(\lambda)$ be a parameter such that $m > N$, and let $\mathcal{X}$ be a set such that there exists a weak PRF family

$$\{H_i : \mathcal{K}_i \times \mathcal{X} \to \mathbb{Z}_q^{m \times m}\}_{i \in [N+1]}.$$

Note that we do not require these weak PRFs to be endowed with any algebraic structure. Finally, we define a family of functions

$$\{F_{i,j} : (\mathbb{Z}_q^{m \times m} \times \mathcal{K}_i \times \mathcal{K}_{j+1}) \times \mathcal{X} \to \mathbb{G}_{i,j}\}_{i,j \in [N], i \le j},$$

where for any $i, j \in [N]$ satisfying $i \le j$, we have

$$F_{i,j}((\mathbf{K}, (k_i, k_{j+1})), x) = g_{i,j}^{H_i(k_i,x) \cdot \mathbf{K}_{i,j} \cdot (H_{j+1}(k_{j+1},x))^{-1}}.$$

Homomorphic Properties. It is easy to see the following:

1. For any $i, j \in [N]$ such that $i \leq j$, for any $x \in \mathcal{X}$, for a fixed (secret) choice of $k_i, k_{j+1} \in \mathcal{K}$ and for any $\mathbf{K}_1, \mathbf{K}_2 \in \mathbb{Z}_q^{m \times m}$, we have

$$F_{i,j}((\mathbf{K}_1 + \mathbf{K}_2, k_i, k_{j+1}), x) = F_{i,j}((\mathbf{K}_1, k_i, k_{j+1}), x) \cdot F_{i,j}((\mathbf{K}_2, k_i, k_{j+1}), x).$$

To see this, observe the following:

$$\begin{aligned} F_{i,j}&((\mathbf{K}_1, k_i, k_{j+1}), x) \cdot F_{i,j}((\mathbf{K}_2, k_i, k_{j+1}), x) \\ &= g_{i,j}^{H(k_i,x)\cdot \mathbf{K}_1 \cdot (H(k_{j+1},x))^{-1}} \cdot g_{i,j}^{H(k_i,x)\cdot \mathbf{K}_2 \cdot (H(k_{j+1},x))^{-1}} \\ &= g_{i,j}^{H(k_i,x)\cdot (\mathbf{K}_1 + \mathbf{K}_2) \cdot (H(k_{j+1},x))^{-1}} \\ &= F_{i,j}((\mathbf{K}_1 + \mathbf{K}_2, k_i, k_{j+1}), x). \end{aligned}$$

2. For any $i, j, \ell \in [N]$ such that $i \leq j < \ell$, for any $x \in \mathcal{X}$, for a fixed (secret) choice of $k_i, k_{j+1}, k_{\ell+1} \in \mathcal{K}$ and for any $\mathbf{K}_1, \mathbf{K}_2 \in \mathbb{Z}_q^{m \times m}$, we have

$$F_{i,\ell}((\mathbf{K}_1 \cdot \mathbf{K}_2, k_i, k_{\ell+1}), x) = e(F_{i,j}((\mathbf{K}_1, k_i, k_{j+1}), x), F_{j+1,\ell}((\mathbf{K}_2, k_{j+1}, k_{\ell+1}), x)).$$

To see this, observe the following:

$$\begin{aligned} e(F_{i,j}&((\mathbf{K}_1, k_i, k_{j+1}), x), F_{j+1,\ell}((\mathbf{K}_2, k_{j+1}, k_{\ell+1}), x)) \\ &= e\left(g_{i,j}^{H(k_i,x)\cdot \mathbf{K}_1 \cdot (H(k_{j+1},x))^{-1}} \quad , \quad g_{j+1,\ell}^{H(k_{j+1},x)\cdot \mathbf{K}_2 (H(k_{\ell+1},x))^{-1}} \right) \\ &= e(g_{i,j}, g_{j+1,\ell})^{H(k_i,x)\mathbf{K}_1 (H(k_{j+1},x))^{-1} \cdot H(k_{j+1},x)\cdot \mathbf{K}_2 \cdot (H(k_{\ell+1},x))^{-1}} \\ &= g_{i,\ell}^{H(k_i,x)\cdot \mathbf{K}_1 \cdot \mathbf{K}_2 \cdot (H(k_{\ell+1},x))^{-1}} \\ &= F_{i,\ell}((\mathbf{K}_1 \cdot \mathbf{K}_2, k_i, k_{\ell+1}), x). \end{aligned}$$

Top-Level Zero Test. It is easy to see that the aforementioned weak PRF family is inherently equipped with a top level zero-test. More specifically, for any $x \in \mathcal{X}$ and any $k_1 \in \mathcal{K}_1$ and any $k_{N+1} \in \mathcal{K}_{N+1}$, we have

$$F_{1,N}((\mathbf{0}, (k_1, k_{N+1})), x) = g_{1,N+1}^{\mathbf{0}},$$

where $\mathbf{0}$ is the all-zero matrx in $\mathbb{Z}_q^{m \times m}$. Thus the weak PRF output on an all-zero subkey is nothing but a matrix whose every entry is the identity element in $\mathbb{G}_{1,N}$, i.e., the target group $\mathbb{G}_T$. This immediately gives an efficiently computable top-level zero test.

Weak PRF Security. We now prove that for any $i, j \in [N]$ such that $i \leq j$, the function $F_{i,j}$ is a weak PRF, under the assumption that the $\mathcal{U}_{n,m}$-matrix DDH (MDDH) assumption holds over the source group $\mathbb{G}$ for sufficiently large $n, m = \text{poly}(\lambda)$, subject to the restriction that $n > m > N$. More formally, we state and prove the following lemma.

Lemma 12.6. *For any $i, j \in [N]$ such that $i \leq j$ and any arbitrarily large $L = \text{poly}(\lambda)$, assuming that:*

- *$\mathcal{U}_{n,m}$-matrix DDH (MDDH) assumption holds over the source group $\mathbb{G}$ for $n = Lm$, and*
- *$H_i : \mathcal{K}_i \times \mathcal{X} \to \mathbb{Z}_q^{m \times m}$ and $H_{j+1} : \mathcal{K}_{j+1} \times \mathcal{X} \to \mathbb{Z}_q^{m \times m}$ are weak PRFs,*

and letting

$$\{x_\ell \leftarrow \mathcal{X}\}_{\ell \in [L]} \quad , \quad (k_i, k_{j+1}) \leftarrow (\mathcal{K}_i \times \mathcal{K}_{j+1}) \quad , \quad \{\mathbf{K} \leftarrow \mathbb{Z}_q^{m \times m}\}_{\ell \in [L]} \quad , \quad \{\mathbf{U}_{\ell,\ell'} \leftarrow \mathbb{Z}_q^{m \times m}\}_{\ell,\ell' \in [L]},$$

we have

$$\begin{bmatrix} F_{i,j}((\mathbf{K}_1,(k_i,k_{j+1})),x_1) & \dots & F_{i,j}((\mathbf{K}_L,(k_i,k_{j+1})),x_1) \\ \vdots & \ddots & \vdots \\ F_{i,j}((\mathbf{K}_1,(k_i,k_{j+1})),x_L) & \dots & F_{i,j}((\mathbf{K}_L,(k_N,k_{N+1})),x_L) \end{bmatrix} \approx_c \begin{bmatrix} g_{i,j}^{\mathbf{U}_{1,1}} & \dots & g_{i,j}^{\mathbf{U}_{1,L}} \\ \vdots & \ddots & \vdots \\ g_{i,j}^{\mathbf{U}_{L,1}} & \dots & g_{i,j}^{\mathbf{U}_{L,L}} \end{bmatrix}$$

Proof. To begin with, we exploit the weak pseudorandomness of the functions H_i amd H_{j+1} to argue that the following holds:

$$\left\{ H_i(k_i,x_\ell), H_{j+1}(k_{j+1},x_\ell) \right\}_{\ell\in[L]} \approx_c \left\{ \mathbf{A}_{\ell,0}, \mathbf{A}_{\ell,1} \right\}_{\ell\in[L]},$$

where for $\ell \in [L]$, $\mathbf{A}_{\ell,0}, \mathbf{A}_{\ell,1} \leftarrow \mathbb{Z}_q^{m\times m}$. This in turn gives us the following:

$$\begin{bmatrix} F_{i,j}((\mathbf{K}_1,(k_i,k_{j+1})),x_1) & \dots & F_{i,j}((\mathbf{K}_L,(k_i,k_{j+1})),x_1) \\ \vdots & \ddots & \vdots \\ F_{i,j}((\mathbf{K}_1,(k_i,k_{j+1})),x_L) & \dots & F_{i,j}((\mathbf{K}_L,(k_N,k_{N+1})),x_L) \end{bmatrix} \approx_c \begin{bmatrix} g_{i,j}^{\mathbf{A}_{1,0}\mathbf{K}_1\mathbf{A}_{1,1}^{-1}} & \dots & g_{i,j}^{\mathbf{A}_{1,0}\mathbf{K}_L\mathbf{A}_{1,1}^{-1}} \\ \vdots & \ddots & \vdots \\ g_{i,j}^{\mathbf{A}_{L,0}\mathbf{K}_1\mathbf{A}_{L,1}^{-1}} & \dots & g_{i,j}^{\mathbf{A}_{L,0}\mathbf{K}_L\mathbf{A}_{L,1}^{-1}} \end{bmatrix}$$

where for $\ell \in [L]$, $\mathbf{A}_{\ell,0}, \mathbf{A}_{\ell,1} \leftarrow \mathbb{Z}_q^{m\times m}$. Next, we invoke Corollary 12.4 to argue the following:

$$\begin{bmatrix} g_{i,j}^{\mathbf{A}_{1,0}\mathbf{K}_1\mathbf{A}_{1,1}^{-1}} & g_{i,j}^{\mathbf{A}_{1,0}\mathbf{K}_2\mathbf{A}_{1,1}^{-1}} & \dots & g_{i,j}^{\mathbf{A}_{1,0}\mathbf{K}_L\mathbf{A}_{1,1}^{-1}} \\ \vdots & \vdots & \ddots & \vdots \\ g_{i,j}^{\mathbf{A}_{L,0}\mathbf{K}_1\mathbf{A}_{L,1}^{-1}} & g_{i,j}^{\mathbf{A}_{L,0}\mathbf{K}_2\mathbf{A}_{L,1}^{-1}} & \dots & g_{i,j}^{\mathbf{A}_{L,0}\mathbf{K}_L\mathbf{A}_{L,1}^{-1}} \end{bmatrix} \approx_c \begin{bmatrix} g_{i,j}^{\mathbf{U}_{1,0}^{(1)}\mathbf{A}_{1,1}^{-1}} & g_{i,j}^{\mathbf{A}_{1,0}\mathbf{K}_2\mathbf{A}_{1,1}^{-1}} & \dots & g_{i,j}^{\mathbf{A}_{1,0}\mathbf{K}_L\mathbf{A}_{1,1}^{-1}} \\ \vdots & \vdots & \ddots & \vdots \\ g_{i,j}^{\mathbf{U}_{1,0}^{(L)}\mathbf{A}_{L,1}^{-1}} & g_{i,j}^{\mathbf{A}_{L,0}\mathbf{K}_2\mathbf{A}_{L,1}^{-1}} & \dots & g_{i,j}^{\mathbf{A}_{L,0}\mathbf{K}_L\mathbf{A}_{L,1}^{-1}} \end{bmatrix},$$

where for each $\ell \in [L]$, we have $\mathbf{U}_{1,0}^{(\ell)} \leftarrow \mathbb{Z}_q^{m\times m}$. Next, we have

$$\begin{bmatrix} g_{i,j}^{\mathbf{U}_{1,0}^{(1)}\mathbf{A}_{1,1}^{-1}} & g_{i,j}^{\mathbf{A}_{1,0}\mathbf{K}_2\mathbf{A}_{1,1}^{-1}} & \dots & g_{i,j}^{\mathbf{A}_{1,0}\mathbf{K}_L\mathbf{A}_{1,1}^{-1}} \\ \vdots & \vdots & \ddots & \vdots \\ g_{i,j}^{\mathbf{U}_{1,0}^{(L)}\mathbf{A}_{L,1}^{-1}} & g_{i,j}^{\mathbf{A}_{L,0}\mathbf{K}_2\mathbf{A}_{L,1}^{-1}} & \dots & g_{i,j}^{\mathbf{A}_{L,0}\mathbf{K}_L\mathbf{A}_{L,1}^{-1}} \end{bmatrix} \approx_s \begin{bmatrix} g_{i,j}^{\mathbf{U}_{1,1}} & g_{i,j}^{\mathbf{A}_{1,0}\mathbf{K}_2\mathbf{A}_{1,1}^{-1}} & \dots & g_{i,j}^{\mathbf{A}_{1,0}\mathbf{K}_L\mathbf{A}_{1,1}^{-1}} \\ \vdots & \vdots & \ddots & \vdots \\ g_{i,j}^{\mathbf{U}_{L,1}} & g_{i,j}^{\mathbf{A}_{L,0}\mathbf{K}_2\mathbf{A}_{L,1}^{-1}} & \dots & g_{i,j}^{\mathbf{A}_{L,0}\mathbf{K}_L\mathbf{A}_{L,1}^{-1}} \end{bmatrix},$$

where for each $\ell \in [L]$, we have $\mathbf{U}_{\ell,1} \leftarrow \mathbb{Z}_q^{m\times m}$. □

The proof of pseudorandomness now follows from a simple hybrid argument, where in hybrid-i we apply the same set of indistinguishable transformations as illustrated above to column i of the original matrix.

Parallel Input Security. Note that unlike all our previous instantiations of slotted weak PRF families where the weak PRF output at each unit interval level $[i,i]$ is completely independent of the weak PRF output at any other interval level $[i',i']$, in this instantiation, the weak PRF output at each unit interval level $[i,i]$ is is related to the weak PRF output at the interval level $[i+1,i+1]$ by the shared H_{i+1} term in the exponent of the generator.

Hence, we prove an additional security property called parallel input security, which states that an ensemble of weak PRF evaluations at each unit interval level on the same input x and using the same inter-level correlated randomness is computationally indistinguishable from uniform. More formally, we state and prove the following lemma.

Lemma 12.7. *For any $N, L = \mathrm{poly}(\lambda)$, assuming that $\{H_i : \mathcal{K}_i \times \mathcal{X} \to \mathbb{Z}_q^{m\times m}\}_{i\in[N+1]}$ is a weak PRF family and letting*

$$x \leftarrow \mathcal{X} \quad , \quad \{k_i \leftarrow \mathcal{K}_i\}_{i\in[N+1]} \quad , \quad \{\mathbf{K}_{\ell,i}, \mathbf{U}_{\ell,i} \leftarrow \mathbb{Z}_q^{m\times m}\}_{\ell\in[L],i\in[N]},$$

we have

$$\begin{bmatrix} F_{1,1}((\mathbf{K}_{1,1},(k_1,k_2)),x) & \dots & F_{N,N}((\mathbf{K}_{1,N},(k_N,k_{N+1})),x) \\ \vdots & \ddots & \vdots \\ F_{1,1}((\mathbf{K}_{L,1},(k_1,k_2)),x) & \dots & F_{N,N}((\mathbf{K}_{L,N},(k_N,k_{N+1})),x) \end{bmatrix} \approx_c \begin{bmatrix} g_{1,1}^{\mathbf{U}_{1,1}} & \dots & g_{N,N}^{\mathbf{U}_{1,N}} \\ \vdots & \ddots & \vdots \\ g_{1,1}^{\mathbf{U}_{L,1}} & \dots & g_{N,N}^{\mathbf{U}_{L,N}} \end{bmatrix}$$

where for each $i \in [N]$, $g_{i,i} = g$ such that $g \leftarrow \mathbb{G}$ is a uniformly sampled generator element.

Proof. First of all, we exploit the weak pseudorandomness of the function family $\{H_i : \mathcal{K}_i \times \mathcal{X} \to \mathbb{Z}_q^{m\times m}\}_{i\in[N+1]}$ to argue that the following holds:

$$\Big\{H_i(k_i,x)\Big\}_{i\in[N+1]} \approx_c \Big\{\mathbf{A}_i\Big\}_{i\in[N+1]},$$

where for each $i \in [N+1]$, $\mathbf{A}_i \leftarrow \mathbb{Z}_q^{m\times m}$. This in turn gives us the following:

$$\begin{bmatrix} F_{1,1}((\mathbf{K}_{1,1},(k_1,k_2)),x) & \dots & F_{N,N}((\mathbf{K}_{1,N},(k_N,k_{N+1})),x) \\ \vdots & \ddots & \vdots \\ F_{1,1}((\mathbf{K}_{L,1},(k_1,k_2)),x) & \dots & F_{N,N}((\mathbf{K}_{L,N},(k_N,k_{N+1})),x) \end{bmatrix} \approx_c \begin{bmatrix} g_{1,1}^{\mathbf{A}_1\mathbf{K}_{1,1}\mathbf{A}_2^{-1}} & \dots & g_{N,N}^{\mathbf{A}_N\mathbf{K}_{1,N}\mathbf{A}_{N+1}^{-1}} \\ \vdots & \ddots & \vdots \\ g_{1,1}^{\mathbf{A}_1\mathbf{K}_{L,1}\mathbf{A}_2^{-1}} & \dots & g_{N,N}^{\mathbf{A}_N\mathbf{K}_{L,N}\mathbf{A}_{N+1}^{-1}} \end{bmatrix},$$

where for each $i \in [N+1]$, $\mathbf{A}_i \leftarrow \mathbb{Z}_q^{m\times m}$. Next, we invoke Lemma 12.5 to argue

$$\begin{bmatrix} g^{\mathbf{A}_1\mathbf{K}_{1,1}\mathbf{A}_2^{-1}} & \dots & g^{\mathbf{A}_N\mathbf{K}_{1,N}\mathbf{A}_{N+1}^{-1}} \\ \vdots & \ddots & \vdots \\ g^{\mathbf{A}_1\mathbf{K}_{L,1}\mathbf{A}_2^{-1}} & \dots & g^{\mathbf{A}_N\mathbf{K}_{L,N}\mathbf{A}_{N+1}^{-1}} \end{bmatrix} \approx_s \begin{bmatrix} g_{1,1}^{\mathbf{U}_{1,1}} & \dots & g_{N,N}^{\mathbf{U}_{1,N}} \\ \vdots & \ddots & \vdots \\ g_{1,1}^{\mathbf{U}_{L,1}} & \dots & g_{N,N}^{\mathbf{U}_{L,N}} \end{bmatrix}.$$

This completes the proof of parallel input security. □

12.3 Constructions from Slotted Partial RKHwPRFs

Note that our constructions from classic RKHwPRFs also work from slotted partial RKHwPRFs. This follows from arguments very similar to those presented in the context of slotted RKHwPRFs. The reader may refer Section 11.3 for a detailed discussion on why the imposition of slots and the corresponding restrictions on the multiplication operations do not hinder our constructions from functioning correctly and securely. In this section, we focus on two key difference between slotted RKHwPRFs and slotted RKHwPRFs, both of which are related to the partial nature of the homomorphism in the latter primitive.

Zero Test. First, unlike slotted RKHwPRFs, a slotted partial RKHwPRF is not inherently equipped with a zero test. The use of additional secret randomness during weak PRF evaluation, and the fact that this randomness does not respect any form of group/ring-homomorphism, implies that evaluating the weak PRF using a zero key is not guaranteed to produce the zero element in the corresponding output interval-level. We take this into account by listing a top-level zero test as an explicit requirement in our definitions of slotted partial RKHwPRF (Definition 12.1). As far as instantiations are concerned, the construction of slotted partial RKHwPRF from matrix-DDH over multilinear maps is naturally equipped with a zero test due to the specific structure of the randomness used during weak PRF evaluation, while the MZ18 MMap is equipped with its own zero test, which can be easily translated into an appropriate zero test for the slotted partial RKHwPRF built from it.

Bounded Hommorphic Operations. Unlike a slotted RKHwPRF where the only restrictions are on the multiplicative depth, a slotted partial RKHwPRF could also be bounded with respect to the number of homomorphic addition operations it supports at any given interval-slot. This is especially relevant with respect to the MZ18 MMap, which builds on top of "noisy" CLT13 encodings, and the noise grows with every additive operation between encodings.

We note however that both our NIKE and iO constructions also have a pre-fixed additive depth. In the case of the NIKE construction, the additive depth of the key derivation circuit is $O(Nm)$, where N is the number of parties participating in the protocol and $m = \mathrm{poly}(\lambda)$ is a fixed matrix-dimension parameter. In the case of the iO construction, the multiplicative depth of the circuit evaluating an obfuscated program is $O((L + N)m)$, where L is the depth of the permutation branching program corresponding to the circuit to be obfuscated, N is the number of input variables and $m = \mathrm{poly}(\lambda)$ is a fixed matrix-dimension parameter of the construction. In particular, for our iO construction, the additive depth of the evaluation circuit is independent of the actual circuit being obfuscated. Hence, as long as the slotted partial RKHwPRF family is securely instantiated with appropriate parameters that supports the requisite number of additively homomorphic operations for our constructions, both correctness and security of our constructions can be ensured.

Rerandomization. Note that in our security proofs for the iO construction from classic RKHwPRFs, we require the ability to create RKHwPRF outputs under fresh keys from a publicly available set of RKHwPRF outputs under random keys. We do this by "subset-summing" over the publicly available RKHwPRF outputs, and then use the leftover hash lemma to argue that the resulting RKHwPRF output is appropriately distributed under a key that is statistically indistinguishable from random.

In the partial RKHwPRF setting, only "subset-summing" over the publicly available RKHwPRF outputs does not necessarily suffice. This is because we need to make sure that the new weak PRF output is not only distributed appropriately with respect to the underlying key, but also the additional underlying randomness. In other words, we need an additional rerandomization algorithm that takes the output of "subset-summing" and re-randomizes it in a manner that preserves both the distribution of the key and the additional randomness (at least in a manner that is computationally indistinguishable from a "freshly created" weak PRF output). This is especially relevant with respect to the MZ18 MMap, which builds on top of "noisy" CLT13 encodings. In particular, "subset-summing" over MZ18 encodings does not necessarily preserve the appropriate noise distribution in the resulting encoding.

However, we note that MZ18 MMap comes equipped with such a rerandomization algorithm. In our security proofs for the MZ18 MMap-based constructions, we thus require only additional step: after "subset-summing" over the publicly available encodings, we use the leftover hash lemma (as in the classic RKHwPRF setting) in conjunction with the MZ18 rerandomization algorithm to make sure that the distribution of the freshly created encoding is computationally indistinguishable from that of a "freshly created" encoding.

The issue of rerandomization does not arise in the slotted partial RKHwPRF construction from multilinear maps where the matrix-DDH assumption holds, due to the special structure of its underlying randomness terms. Here, we can simply "subset-sum" over existing weak PRF outputs and argue that the resulting RKHwPRF output is appropriately distributed under a key that is statistically indistinguishable from random. The argument uses the leftover hash lemma and follows exactly as in the classic RKHwPRF case.

13 Impossibility of Field(-embedded) Homomorphic Synthesizers

In this section, we show that there is no (secure) Field-embedded Homomorphic Synthesizer (FHS). Since a field-embedded homomorphic synthesizer is trivially implied by a Field KHwPRF (or a Field-homomorphic Synthesizer), it follows that there is no (secure) Field KHwPRF (or a Field-homomorphic Synthesizer) as well. First, we define the notion of a field-embedded homomorphic synthesizer:

Definition 13.1. (Field-embedded Homomorphic Synthesizer.) A Field-embedded Homomorphic Synthesizer (FHS) $S : X \times G \to F$ is a synthesizer that satisfies the following properties:

- $(G, \oplus)$ is an efficiently samplable (finite) group with efficiently computable group operation.
- $(F, \boxplus, \boxtimes)$ is an efficiently samplable (finite) field with efficiently computable field operations.
- For any $x \in X$ the function $S(x, \cdot) : G \to F$ is a *group* homomorphism, i.e., for any $x \in X$ and $f_1, f_2 \in F$ we have
$$S(x, g_1 \oplus g_2) = S(x, f_1) \boxplus S(x, f_2)$$

It is easy to see that a ring-embedded homomorphic synthesizer is implied by an RKHwPRF or an RIHwPRF (for which the input space does not depend on the choice of the key).

Informally, a field-embedded homomorphic synthesizer is a stronger version of ring-embedded homomorphic synthesizer which enables to efficiently compute the inverse in the output field. We now show that there is a (generic) attack against any field-embedded homomorphic synthesizer.

Let $S : X \times \bar{F} \to F$ be a field-embedded homomorphic synthesizer, and fix an integer $m > 3 \log |\bar{F}|$. If $\mathbf{F} \leftarrow F^{m \times m}$ and $\mathbf{s} \leftarrow \{0_F, 1_F\}^m$, by Theorem 4.1 it follows that

$$(\mathbf{F}, \mathbf{F}\mathbf{s}) \overset{c}{\approx} (\mathbf{F}, \mathbf{u}),$$

where $\mathbf{u} \leftarrow F^m$ is a uniformly chosen vector of field elements. We define the set $\mathcal{S}$ as

$$\mathcal{S} = \{\mathbf{F}\mathbf{s} : \mathbf{s} \in \{0_F, 1_F\}\}$$

Since $|F|$ is superpolynomially large in λ (otherwise the attack is straightforward), it follows that

- $\mathbf{F}$ is a full-rank matrix with high probability.
- $\Pr[\mathbf{u} \in \mathcal{S}] \leq \mathrm{negl}(\lambda)$ where the probability is taken over the randomness of $\mathbf{F}$ and $\mathbf{u}$.

Given a pair of the form $(\mathbf{F}, \mathbf{c})$ where either $\mathbf{c} = \mathbf{F}\mathbf{s}$ or $\mathbf{c}$ is a uniform vector over F^m, the attacker solves the (linear) equation $\mathbf{F}\mathbf{x} = \mathbf{c}$ and checks whether the solution is binary. Notice that Gaussian elimination is possible since the field operations (including inverse) can be efficiently done in F. If there exists a binary solution, the attacker outputs 1. Otherwise, it outputs 0. It is easy to see that the advantage of the attacker in distinguishing $(\mathbf{F}, \mathbf{F}\mathbf{s})$ and $(\mathbf{F}, \mathbf{u})$ is close to 1. Therefore, there is no (secure) field-embedded homomorphic synthesizer.

14 Conclusion and Future Work

In this paper, we showed how to build iO and multiparty NIKE from a ring-embedded homomorphic synthesizer. In particular, our iO construction is secure in the standard model and from a program independent assumption. We think that RHS (or RKHwPRF) is one of the simplest and most understandable primitives that is currently known to directly imply iO. We also showed that asymmetric multilinear maps that satisfy certain properties imply RKHwPRF, which in turn imply RHS.

Public Key Cryptography and Mathematical Structure. This work also mostly completes the line of work started in [AMPR19] that suggests that public-key cryptographic primitives inherently follow from structured Minicrypt primitives. With this work, we can show that all of the most common cryptosystems can, in fact, be built using a structured Minicrypt primitive. The structure over a Minicrypt primitive also happens to be easy to state: either a group or ring homomorphism over the input space or key space.

This bolsters the argument that it makes sense to base theoretical constructions of cryptosystems (i.e., constructions that are focused on showing the existence of something rather than a practical implementation) on generic primitives rather than concrete assumptions, since the generic primitives protect against specific assumptions being broken. We defer to the work of [AMPR19] for a more eloquent argument of this point.

The intuition about structure and public-key cryptography in this work can also be applied to some primitives that have not been studied before. For instance, threshold signatures using bilinear maps (e.g. as in [BLS01]-based signatures) exploit the full structure of a field in order to use Shamir secret sharing [Sha79]. Lattice-based assumptions typically are only (bounded) ring homomorphic (and not field homomorphic). This is perhaps a natural reason why lattice-based threshold signature constructions [BGG$^+$18] are seemingly harder to construct than those from bilinear maps, and finding a way to build a field homomorphic primitive using lattices might be a plausible way to try build efficient threshold signatures from lattices.

Constructing an RKHwPRF. Currently, we know of no existing, provably secure constructions of an RKHwPRF from standard assumptions. However, it is certainly worth it to briefly discuss some strategies for constructing an RKHwPRF and why known techniques do not work. A very natural starting question that an ambitious researcher might ask is the following: is it possible to use the highly structured primitives from [AMPR19] to build iO? This question is particularly interesting because RIHwPRFs with certain properties are known to be implied by FHE, which is in turn implied by standard LWE.

From [GMM17], we know that building iO from FHE is impossible in a certain black-box sense. This means that it is unlikely that we can take a generic IHwPRF and use it to construct a KHwPRF. Concretely, [AMPR19] showed that equipping weak PRFs with a ring homomorphism over the input space (RIHwPRF) with certain other requirements yields a fully homomorphic encryption scheme. In addition, FHE also implies an RIHwPRF with certain properties. On the other hand, our results here show that RKHwPRFs imply the powerful notion of indistinguishability obfuscation. So we might ask, what are the qualitative differences between these two constructions? And what is the intuitive reason why we cannot construct RKHwPRFs from RIHwPRFs? If we closely examine the construction of RIHwPRF from FHE in [AMPR19], we see that the resulting RIHwPRFs have a substantial drawback that prevents us from building an RKHwPRF: the output space of the RIHwPRF is dependent on the choice of the key.

References

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

[ADM06] V. Arvind, B. Das, and P. Mukhopadhyay. The complexity of black-box ring problems. In *Proceedings of the 12th Annual International Conference on Computing and Combinatorics*, COCOON'06, pages 126–135. Springer-Verlag, Berlin, Heidelberg, 2006. ISBN 3-540-36925-2, 978-3-540-36925-7.

[AFH$^+$16] M. R. Albrecht, P. Farshim, D. Hofheinz, E. Larraia, and K. G. Paterson. Multilinear maps from obfuscation. In E. Kushilevitz and T. Malkin, editors, *TCC 2016-A, Part I*, volume 9562 of *LNCS*, pages 446–473. Springer, Heidelberg, January 2016.

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

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

[AJL$^+$19] P. Ananth, A. Jain, H. Lin, C. Matt, and A. Sahai. Indistinguishability obfuscation without multilinear maps: New paradigms via low degree weak pseudorandomness and security amplification. In A. Boldyreva and D. Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 284–332. Springer, Heidelberg, August 2019.

[Alb19] M. Albrecht. Are graded encoding schemes broken yet?, 2019.

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

[AMP19] N. Alamati, H. Montgomery, and S. Patranabis. Symmetric primitives with structured secrets. In A. Boldyreva and D. Micciancio, editors, *CRYPTO 2019, Part I*, volume 11692 of *LNCS*, pages 650–679. Springer, Heidelberg, August 2019.

[AMP20] N. Alamati, H. Montgomery, and S. Patranabis. Stronger multilinear maps from indistinguishability obfuscation. Cryptology ePrint Archive, Report 2020/610, 2020.

[AMPR19] N. Alamati, H. Montgomery, S. Patranabis, and A. Roy. Minicrypt primitives with algebraic structure and applications. In Y. Ishai and V. Rijmen, editors, *EUROCRYPT 2019, Part II*, volume 11477 of *LNCS*, pages 55–82. Springer, Heidelberg, May 2019.

[AS17] P. Ananth and A. Sahai. Projective arithmetic functional encryption and indistinguishability obfuscation from degree-5 multilinear maps. In J. Coron and J. B. Nielsen, editors, *EUROCRYPT 2017, Part I*, volume 10210 of *LNCS*, pages 152–181. Springer, Heidelberg, April / May 2017.

[Bar17] B. Barak. The complexity of public-key cryptography. In *Tutorials on the Foundations of Cryptography*, pages 45–77. 2017.

[BBKK18] B. Barak, Z. Brakerski, I. Komargodski, and P. K. Kothari. Limits on low-degree pseudorandom generators (or: Sum-of-squares meets program obfuscation). In J. B. Nielsen and V. Rijmen, editors, *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 649–679. Springer, Heidelberg, April / May 2018.

[BDGM20] Z. Brakerski, N. Döttling, S. Garg, and G. Malavolta. Candidate iO from homomorphic encryption schemes. In V. Rijmen and Y. Ishai, editors, *EUROCRYPT 2020, Part I*, LNCS, pages 79–109. Springer, Heidelberg, May 2020.

[BF01] D. Boneh and M. K. Franklin. Identity-based encryption from the Weil pairing. In J. Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 213–229. Springer, Heidelberg, August 2001.

[BGG$^+$18] D. Boneh, R. Gennaro, S. Goldfeder, A. Jain, S. Kim, P. M. R. Rasmussen, and A. Sahai. Threshold cryptosystems from threshold fully homomorphic encryption. In H. Shacham and A. Boldyreva, editors, *CRYPTO 2018, Part I*, volume 10991 of *LNCS*, pages 565–596. Springer, Heidelberg, August 2018.

[BGI$^+$01] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. P. Vadhan, and K. Yang. On the (im)possibility of obfuscating programs. In J. Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 1–18. Springer, Heidelberg, August 2001.

[BGK$^+$14] B. Barak, S. Garg, Y. T. Kalai, O. Paneth, and A. Sahai. Protecting obfuscation against algebraic attacks. In P. Q. Nguyen and E. Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 221–238. Springer, Heidelberg, May 2014.

[BGMZ18] J. Bartusek, J. Guan, F. Ma, and M. Zhandry. Return of GGH15: Provable security against zeroizing attacks. In A. Beimel and S. Dziembowski, editors, *TCC 2018, Part II*, volume 11240 of *LNCS*, pages 544–574. Springer, Heidelberg, November 2018.

[BKM17] D. Boneh, S. Kim, and H. W. Montgomery. Private puncturable PRFs from standard lattice assumptions. In J. Coron and J. B. Nielsen, editors, *EUROCRYPT 2017, Part I*, volume 10210 of *LNCS*, pages 415–445. Springer, Heidelberg, April / May 2017.

[BKM$^+$19] A. Bishop, L. Kowalczyk, T. Malkin, V. Pastro, M. Raykova, and K. Shi. In pursuit of clarity in obfuscation. Cryptology ePrint Archive, Report 2019/463, 2019. https://eprint.iacr.org/2019/463.

[BLMR13] D. Boneh, K. Lewi, H. W. Montgomery, and A. Raghunathan. Key homomorphic PRFs and their applications. In R. Canetti and J. A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 410–428. Springer, Heidelberg, August 2013.

[BLS01] D. Boneh, B. Lynn, and H. Shacham. Short signatures from the Weil pairing. In C. Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 514–532. Springer, Heidelberg, December 2001.

[BMSZ15] S. Badrinarayanan, E. Miles, A. Sahai, and M. Zhandry. Post-zeroizing obfuscation: The case of evasive circuits. Cryptology ePrint Archive, Report 2015/167, 2015. http://eprint.iacr.org/2015/167.

[BP15] N. Bitansky and O. Paneth. ZAPs and non-interactive witness indistinguishability from indistinguishability obfuscation. In Y. Dodis and J. B. Nielsen, editors, *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 401–427. Springer, Heidelberg, March 2015.

[BR14a] Z. Brakerski and G. N. Rothblum. Black-box obfuscation for d-CNFs. In M. Naor, editor, *ITCS 2014*, pages 235–250. ACM, January 2014.

[BR14b] Z. Brakerski and G. N. Rothblum. Virtual black-box obfuscation for all circuits via generic graded encoding. In Y. Lindell, editor, *TCC 2014*, volume 8349 of *LNCS*, pages 1–25. Springer, Heidelberg, February 2014.

[Bra12] Z. Brakerski. Fully homomorphic encryption without modulus switching from classical GapSVP. In R. Safavi-Naini and R. Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 868–886. Springer, Heidelberg, August 2012.

[BV11] Z. Brakerski and V. Vaikuntanathan. Fully homomorphic encryption from ring-LWE and security for key dependent messages. In P. Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 505–524. Springer, Heidelberg, August 2011.

[BV15] N. Bitansky and V. Vaikuntanathan. Indistinguishability obfuscation from functional encryption. In V. Guruswami, editor, *56th FOCS*, pages 171–190. IEEE Computer Society Press, October 2015.

[BZ14] D. Boneh and M. Zhandry. Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. In J. A. Garay and R. Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 480–499. Springer, Heidelberg, August 2014.

[CC17] R. Canetti and Y. Chen. Constraint-hiding constrained PRFs for NC^1 from LWE. In J. Coron and J. B. Nielsen, editors, *EUROCRYPT 2017, Part I*, volume 10210 of *LNCS*, pages 446–476. Springer, Heidelberg, April / May 2017.

[CCH$^+$19] J. H. Cheon, W. Cho, M. Hhan, J. Kim, and C. Lee. Statistical zeroizing attack: Cryptanalysis of candidates of BP obfuscation over GGH15 multilinear map. In A. Boldyreva and D. Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 253–283. Springer, Heidelberg, August 2019.

[CGH17] Y. Chen, C. Gentry, and S. Halevi. Cryptanalyses of candidate branching program obfuscators. In J. Coron and J. B. Nielsen, editors, *EUROCRYPT 2017, Part III*, volume 10212 of *LNCS*, pages 278–307. Springer, Heidelberg, April / May 2017.

[CHL$^+$15] J. H. Cheon, K. Han, C. Lee, H. Ryu, and D. Stehlé. Cryptanalysis of the multilinear map over the integers. In E. Oswald and M. Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 3–12. Springer, Heidelberg, April 2015.

[CLLT16] J.-S. Coron, M. S. Lee, T. Lepoint, and M. Tibouchi. Cryptanalysis of GGH15 multilinear maps. In M. Robshaw and J. Katz, editors, *CRYPTO 2016, Part II*, volume 9815 of *LNCS*, pages 607–628. Springer, Heidelberg, August 2016.

[CLLT17] J.-S. Coron, M. S. Lee, T. Lepoint, and M. Tibouchi. Zeroizing attacks on indistinguishability obfuscation over CLT13. In S. Fehr, editor, *PKC 2017, Part I*, volume 10174 of *LNCS*, pages 41–58. Springer, Heidelberg, March 2017.

[CLT13] J.-S. Coron, T. Lepoint, and M. Tibouchi. Practical multilinear maps over the integers. In R. Canetti and J. A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 476–493. Springer, Heidelberg, August 2013.

[CLT15] J.-S. Coron, T. Lepoint, and M. Tibouchi. New multilinear maps over the integers. In R. Gennaro and M. J. B. Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 267–286. Springer, Heidelberg, August 2015.

[DH76] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, 1976.

[EHK+13] A. Escala, G. Herold, E. Kiltz, C. Ràfols, and J. Villar. An algebraic framework for Diffie-Hellman assumptions. In R. Canetti and J. A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 129–147. Springer, Heidelberg, August 2013.

[FHHL18] P. Farshim, J. Hesse, D. Hofheinz, and E. Larraia. Graded encoding schemes from obfuscation. In M. Abdalla and R. Dahab, editors, *PKC 2018, Part II*, volume 10770 of *LNCS*, pages 371–400. Springer, Heidelberg, March 2018.

[Gen09] C. Gentry. Fully homomorphic encryption using ideal lattices. In M. Mitzenmacher, editor, *41st ACM STOC*, pages 169–178. ACM Press, May / June 2009.

[GGH13a] S. Garg, C. Gentry, and S. Halevi. Candidate multilinear maps from ideal lattices. In T. Johansson and P. Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 1–17. Springer, Heidelberg, May 2013.

[GGH+13b] S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, and B. Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th FOCS*, pages 40–49. IEEE Computer Society Press, October 2013.

[GGH15] C. Gentry, S. Gorbunov, and S. Halevi. Graph-induced multilinear maps from lattices. In Y. Dodis and J. B. Nielsen, editors, *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 498–527. Springer, Heidelberg, March 2015.

[GGHR14] S. Garg, C. Gentry, S. Halevi, and M. Raykova. Two-round secure MPC from indistinguishability obfuscation. In Y. Lindell, editor, *TCC 2014*, volume 8349 of *LNCS*, pages 74–94. Springer, Heidelberg, February 2014.

[GLSW15] C. Gentry, A. B. Lewko, A. Sahai, and B. Waters. Indistinguishability obfuscation from the multilinear subgroup elimination assumption. In V. Guruswami, editor, *56th FOCS*, pages 151–170. IEEE Computer Society Press, October 2015.

[GMM+16] S. Garg, E. Miles, P. Mukherjee, A. Sahai, A. Srinivasan, and M. Zhandry. Secure obfuscation in a weak multilinear map model. In M. Hirt and A. D. Smith, editors, *TCC 2016-B, Part II*, volume 9986 of *LNCS*, pages 241–268. Springer, Heidelberg, October / November 2016.

[GMM17] S. Garg, M. Mahmoody, and A. Mohammed. Lower bounds on obfuscation from all-or-nothing encryption primitives. In J. Katz and H. Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 661–695. Springer, Heidelberg, August 2017.

[GSW13] C. Gentry, A. Sahai, and B. Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In R. Canetti and J. A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 75–92. Springer, Heidelberg, August 2013.

[HB15] M. Horváth and L. Buttyán. The birth of cryptographic obfuscation-a survey. Technical report, Cryptology ePrint Archive, Report 2015/412, 2015.

[HJ16] Y. Hu and H. Jia. Cryptanalysis of GGH map. In M. Fischlin and J.-S. Coron, editors, *EUROCRYPT 2016, Part I*, volume 9665 of *LNCS*, pages 537–565. Springer, Heidelberg, May 2016.

[HSW14] S. Hohenberger, A. Sahai, and B. Waters. Replacing a random oracle: Full domain hash from indistinguishability obfuscation. In P. Q. Nguyen and E. Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 201–220. Springer, Heidelberg, May 2014.

[IZ89] R. Impagliazzo and D. Zuckerman. How to recycle random bits. In *30th FOCS*, pages 248–253. IEEE Computer Society Press, October / November 1989.

[Jag12] T. Jager. *On black-box models of computation in cryptology*. Ph.D. thesis, Ruhr University Bochum, 2012.

[JLMS19] A. Jain, H. Lin, C. Matt, and A. Sahai. How to leverage hardness of constant-degree expanding polynomials overa $\mathbb{R}$ to build $i\mathcal{O}$. In Y. Ishai and V. Rijmen, editors, *EUROCRYPT 2019, Part I*, volume 11476 of *LNCS*, pages 251–281. Springer, Heidelberg, May 2019.

[Jou00] A. Joux. A one round protocol for tripartite diffie–hellman. In *International algorithmic number theory symposium*, pages 385–393. Springer, 2000.

[Kil88] J. Kilian. Founding cryptography on oblivious transfer. In *20th ACM STOC*, pages 20–31. ACM Press, May 1988.

[Lin16] H. Lin. Indistinguishability obfuscation from constant-degree graded encoding schemes. In M. Fischlin and J.-S. Coron, editors, *EUROCRYPT 2016, Part I*, volume 9665 of *LNCS*, pages 28–57. Springer, Heidelberg, May 2016.

[Lin17] H. Lin. Indistinguishability obfuscation from SXDH on 5-linear maps and locality-5 PRGs. In J. Katz and H. Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 599–629. Springer, Heidelberg, August 2017.

[LT17] H. Lin and S. Tessaro. Indistinguishability obfuscation from trilinear maps and block-wise local PRGs. In J. Katz and H. Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 630–660. Springer, Heidelberg, August 2017.

[LV16] H. Lin and V. Vaikuntanathan. Indistinguishability obfuscation from DDH-like assumptions on constant-degree graded encodings. In I. Dinur, editor, *57th FOCS*, pages 11–20. IEEE Computer Society Press, October 2016.

[LV17] A. Lombardi and V. Vaikuntanathan. Limits on the locality of pseudorandom generators and applications to indistinguishability obfuscation. In Y. Kalai and L. Reyzin, editors, *TCC 2017, Part I*, volume 10677 of *LNCS*, pages 119–137. Springer, Heidelberg, November 2017.

[Mic02] D. Micciancio. Generalized compact knapsacks, cyclic lattices, and efficient one-way functions from worst-case complexity assumptions. In *43rd FOCS*, pages 356–365. IEEE Computer Society Press, November 2002.

[MMN16] M. Mahmoody, A. Mohammed, and S. Nematihaji. On the impossibility of virtual black-box obfuscation in idealized models. In E. Kushilevitz and T. Malkin, editors, *TCC 2016-A, Part I*, volume 9562 of *LNCS*, pages 18–48. Springer, Heidelberg, January 2016.

[MR07] U. M. Maurer and D. Raub. Black-box extension fields and the inexistence of field-homomorphic one-way permutations. In K. Kurosawa, editor, *ASIACRYPT 2007*, volume 4833 of *LNCS*, pages 427–443. Springer, Heidelberg, December 2007.

[MSS16] A. Menezes, P. Sarkar, and S. Singh. Challenges with assessing the impact of nfs advances on the security of pairing-based cryptography. In *International Conference on Cryptology in Malaysia*, pages 83–108. Springer, 2016.

[MSZ16] E. Miles, A. Sahai, and M. Zhandry. Annihilation attacks for multilinear maps: Cryptanalysis of indistinguishability obfuscation over GGH13. In M. Robshaw and J. Katz, editors, *CRYPTO 2016, Part II*, volume 9815 of *LNCS*, pages 629–658. Springer, Heidelberg, August 2016.

[MZ18] F. Ma and M. Zhandry. The MMap strikes back: Obfuscation and new multilinear maps immune to CLT13 zeroizing attacks. In A. Beimel and S. Dziembowski, editors, *TCC 2018, Part II*, volume 11240 of *LNCS*, pages 513–543. Springer, Heidelberg, November 2018.

[NPR99] M. Naor, B. Pinkas, and O. Reingold. Distributed pseudo-random functions and KDCs. In J. Stern, editor, *EUROCRYPT'99*, volume 1592 of *LNCS*, pages 327–346. Springer, Heidelberg, May 1999.

[NR95] M. Naor and O. Reingold. Synthesizers and their application to the parallel construction of pseudo-random functions. In *36th FOCS*, pages 170–181. IEEE Computer Society Press, October 1995.

[Ps16] R. Pass and a. shelat. Impossibility of VBB obfuscation with ideal constant-degree graded encodings. In E. Kushilevitz and T. Malkin, editors, *TCC 2016-A, Part I*, volume 9562 of *LNCS*, pages 3–17. Springer, Heidelberg, January 2016.

[Reg05] O. Regev. On lattices, learning with errors, random linear codes, and cryptography. In H. N. Gabow and R. Fagin, editors, *37th ACM STOC*, pages 84–93. ACM Press, May 2005.

[RSA78] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[Sha79] A. Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[Sho97] V. Shoup. Lower bounds for discrete logarithms and related problems. In W. Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 256–266. Springer, Heidelberg, May 1997.

[SW14] A. Sahai and B. Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In D. B. Shmoys, editor, *46th ACM STOC*, pages 475–484. ACM Press, May / June 2014.

[YYHK18] T. Yamakawa, S. Yamada, G. Hanaoka, and N. Kunihiro. Generic hardness of inversion on ring and its relation to self-bilinear map. Cryptology ePrint Archive, Report 2018/463, 2018. `https://eprint.iacr.org/2018/463`.

[Zim15] J. Zimmerman. How to obfuscate programs directly. In E. Oswald and M. Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 439–467. Springer, Heidelberg, April 2015.