Indistinguishability Obfuscation from Ring Key-Homomorphic Weak PRFs

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. A recent work by Alamati *et al*. (CT-RSA' 23) initiated the study of ring key-homomorphic weak PRFs (RKHwPRFs) and showed that any RKHwPRF can be used to construct multiparty noninteractive key exchange (NIKE) for an arbitrary number of parties. In this work, we show that *any* RKHwPRF can, in fact, be used to construct indistinguishability obfuscation (iO) for all circuits in $\mathcal{NC}^1$, which in turn can be bootstrapped to all polynomial-size circuits using standard techniques. The proof of security for our iO construction is in the standard model, and our assumptions (including weakenings of RKHwPRFs) are program-independent.

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, notably 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 result in a sense completes the work initiated by Alamati *et al.* (Eurocrypt' 19, JoC '23) on building cryptosystems from generic Minicrypt primitives with structure. Given our construction of iO from RKHwPRFs, 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.

*Linux Foundation.
†IBM Research India.

Contents

1 Introduction **3**
- 1.1 Structure and Cryptography . . . 4
- 1.2 Our Contributions . . . 5
- 1.3 Discussion . . . 7

2 Technical Overview **9**
- 2.1 Definitions and Technical Tools . . . 9
- 2.2 Overview of Our Approach . . . 11
- 2.3 A Simplified Version of Our iaiO Construction . . . 15
- 2.4 Security Proof Intuition . . . 19
- 2.5 A Sufficient Ring Assumption for Our iaiO Construction . . . 21
- 2.6 Relaxations of Classic RKHwPRFs . . . 22
- 2.7 iO from "The MMap Strikes Back" . . . 24

3 Preliminaries **28**
- 3.1 Notation . . . 28
- 3.2 Cryptographic Primitives . . . 29
- 3.3 Leftover Hash Lemma . . . 34
- 3.4 Some Results on RHS . . . 34

4 Input-Activated Indistinguishability Obfuscator from RHS **36**
- 4.1 Core iO Construction . . . 36
- 4.2 Parallel iO Construction . . . 41
- 4.3 Incorporating Input-Activations: iaiO Construction . . . 44
- 4.4 Summary: The Final iaiO Construction . . . 45
- 4.5 Proof of Correctness . . . 48

5 Security from Subspace Hiding Assumptions **50**
- 5.1 Notations . . . 50
- 5.2 The Subspace Hiding Assumptions . . . 52
- 5.3 The Main Theorems . . . 55

6 Proof of Theorem 5.5 **57**
- 6.1 Inter-Column Security . . . 57
- 6.2 Single-Input Program Switching Security . . . 65
- 6.3 Completely Inactive Program Security and Intra-Column Security . . . 71

7 Proof Of Theorem 5.6 **72**
- 7.1 Lifting Lemma . . . 72
- 7.2 From "Baby" Subspace Hiding Assumption To Full Assumption . . . 78
- 7.3 An Auxiliary Subspace Hiding Assumption . . . 81
- 7.4 Proof of Baby Subspace Hiding Assumption-1 . . . 94
- 7.5 Proof of Baby Subspace Hiding Assumption-2 . . . 102

8 Relaxations of Classic RKHwPRFs **107**
- 8.1 Slotted RKHwPRFs . . . 107
- 8.2 Slotted Partial RKHwPRFs . . . 110
- 8.3 Partial Slotted RKHwPRF from the MZ18 MMap . . . 117

9 Conclusion and Future Work **121**

A Some Basic Lemmata 130

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]. This has subsequently been followed by the development of pairing-based cryptography [Jou04, BF01] and lattice-based cryptography [Reg05, Gen09], both of which enabled applications not known from previously studied assumptions.

In 2013, Garg, Gentry, and Halevi proposed the first candidate multilinear map construction [GGH13a] which gained a lot of attention in 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 successfully attacked, 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, researchers 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 [Alb19].

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. These constructions cleverly leveraged functional encryption [BSW11] to build iO. 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.[2] These constructions provide many new insights, and seem to have moved us closer to the "holy grail" of constructing iO from standard cryptographic assumptions. Bishop *et al.* summarized the sentiment of the community on the wide range of assumptions used to build iO in [BKM+19].

More Recent Work. After a period of considerable skepticism about the feasibility of iO, there were some breakthrough results on new iO constructions. Brakerski *et al.* showed how to build iO from a new primitive called *split FHE* [BDGM20a]. Their construction relies on a plausible but heuristic security argument. Subsequently, Gay and Pass [GP21] built a version of the [BDGM20a] scheme that relies on the learning with errors (LWE) assumption [Reg05], the decisional composite residuosity assumption [DJ01], and a circular security conjecture on the Gentry-Sahai-Waters FHE scheme [GSW13] and the Damgard-Jurik encryption scheme [DJ01]. Brakerski *et al.* further improved the Gay and Pass construction so that security only relied on a variant of circular-secure LWE [BDGM20b]. Wee and Wichs [WW21] also had a follow-up work building iO from split FHE. Unfortunately, a recent work by Hopkins *et al.* [HJL21] showed certain counterexamples to some of the core assumptions in [BDGM20b, WW21, GP21], although this result does not immediately break the iO constructions in these papers themselves.

In one of the most exciting recent works in all of cryptography, Jain *et al.* [JLS21] published a new breakthrough construction of iO from well-founded assumptions, including LWE, the symmetric external Diffie-Hellman assumption

[1] Graded encodings are generalizations of multilinear maps.

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

(SXDH) [EHK+13], a version of the learning parity with noise (LPN) assumption [Ale03] over a large modulus, and the existence of a boolean PRG in NC^0 with superlinear stretch. Jain *et al.* [JLS22] also followed up this work with a new, similar construction that did not rely on LWE, eliminating the need for lattice-based assumptions. In addition, Gay *et al.* [GJLS21] studied how to build iO from simpler assumptions. There has been additional work aiming at constructing iO from plausibly post-quantum assumptions [DQV+21], but this was later found to have used a assumption that was insecure [JLLS23]. Most recently, Ragavan *et al.* [RVV24] showed how to construct iO from bilinear maps and LPN variants, eliminating the need for the boolean PRGs.

These recent results indicate that iO is realizable from relatively standard assumptions. At the same time, they rely on very specific assumptions and proof techniques that are quite different from each other. We believe that it is still very useful to study what sort of generic mathematical structure and assumptions are sufficient for iO–and in fact, we think these exciting recent results motivate such a study further–as it may lead to better understanding of iO itself, as well as new constructions of iO from potentially weaker assumptions.

Our Approach. In this paper, we study iO in a way that differs from the most common previous approach of directly proposing more assumptions and constructions (although we certainly agree that this traditional approach has plenty of merit, as the recent breakthrough results attest). 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 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 simple assumptions (such as "vanilla" LWE). 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. For instance, Barak [Bar17] has ruminated on this topic in detail. This idea was formalized to a degree in a work by Alamati *et al.* [AMPR19] that focused on the relationship between mathematical structure and cryptographic functionality.

Primitives with Structured Inputs. 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 not completely satisfactory since they almost mimic multilinear maps.

Primitives with Structured Secrets. 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 iO.

In a more recent work [AMP23], Alamati *et al.* introduced new, simple primitives called *ring key-homomorphic weak pseudorandom functions* (RKHwPRF) and *ring-embedded homomorphic synthesizers* (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 provide the formal definitions of these primitives later in the paper). In [AMP23], Alamati *et al.* showed how to build multiparty noninteractive key exchange (NIKE) from any RHS (and hence from any RKHwPRF). While iO implies multiparty NIKE [BZ14], the reverse implication is unknown.

iO from Structured Primitives? Despite all of the recent work studying mathematical structure in cryptography, there is very little to show on iO. 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 assumptions necessary for secure iO. This motivates us to ask the following question:

Are there simple, algebraically structured primitives that imply iO?

Answering this question would provide more insight into 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 show how to build iO from a generic RHS, and hence any generic RKHwPRF (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.

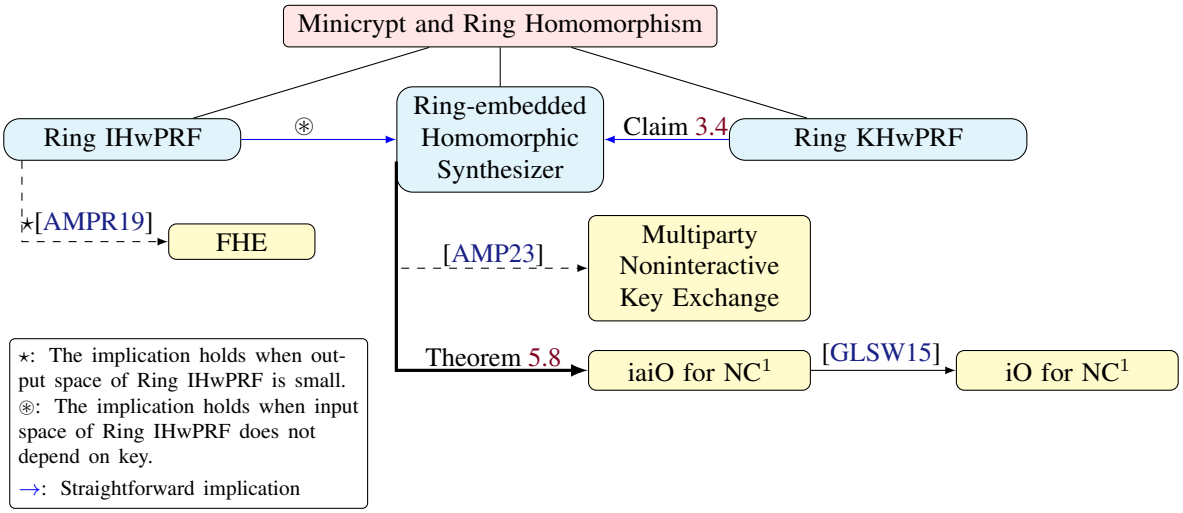

Figure 1: Summary of the Implications of Symmetric Primitives with Ring Homomorphism

Building (ia)iO from RHS. Our main result is a construction of iO (for $\mathcal{NC}^1$) from a ring-embedded homomorphic synthesizer. Our 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.

More concretely, we show how to build *input-activated* iO (iaiO) from any RHS, which in turn implies standard iO due to [GLSW15]. Informally speaking, iaiO allows starting with an initial set of programs and then adding modifications so that the program can be "turned off" on an arbitrary set of inputs. While iaiO is seemingly more complicated primitive than regular iO, it is considerably easier to achieve from simple, easy-to-state assumptions (and this is presumably by design in [GLSW15]). Our construction of iaiO 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].

We develop many new techniques for dealing with non-commutative 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.

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

iO from "The MMap Strikes Back" [MZ18] 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.[1] 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. In particular, they leave constructing iO from their multilinear map 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 to show that such a graded encoding does, in fact, imply a (slotted, partial) RKHwPRF, which means we can get an iO construction.

Post-Quantum (In)Security of Our Construction. Unfortunately, the [MZ18] multilinear map (and our construction of slotted partial RKHwPRF from it) are *not post-quantum secure*. In particular, the [MZ18] multilinear map builds upon the [CLT13] multilinear map, and thus any hardness assumption over the [MZ18] multilinear map is broken trivially by quantum attacks that factor the [CLT13] modulus (see discussion on Pg-10 of [MZ18]). Hence, we do not obtain a plausibly post-quantum secure construction of iO in this paper.

We do emphasize that there is no fundamental reason why an RHS/RKHwPRF with a bounded number of homomorphic operations cannot be post-quantum secure. We think that constructing a candidate post-quantum RHS/RKHwPRF would be good potential way to try to build post-quantum iO. We leave this as a very interesting open question.

New Separation Results. Our findings also yield new and interesting black-box 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, [AMPR19] shows 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.

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. These results advance our understanding of the relationship between iO and multilinear maps, but unfortunately cannot be used with our work to show the equivalence of iO and multilinear maps, since we do not know how to build iO from a multilinear map where only the MLDDH assumption holds. In fact, it seems quite challenging to build iO from multilinear maps with only "target" group assumptions.

While we unfortunately do not completely close this gap, our work here makes progress on this problem of building "powerful" multilinear maps from iO by weakening the assumptions over multilinear maps needed to construct iO. To do this, we consider restricted versions of RKHwPRFs that are structurally weaker than a classic RKHwPRF but suffice for construction of iO. We show how to instantiate these restricted RKHwPRFs from various multilinear maps and associated assumptions. Our framework gives several new results, such as: (i) the first iO scheme that relies *only* on SXDH over any asymmetric multilinear map without additional assumptions, (ii) the first iO scheme that relies *only* on DLIN (or more generally Matrix-DDH) over any (even symmetric) multilinear map without additional assumptions, and (iii) the first iO scheme that relies on SXDH over the multilinear map presented by Ma and Zhandry (TCC' 18) (the

[1] 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.

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. We refer to Section 2.6 for details.

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 of iaiO for $\mathcal{NC}^1$ from RHS. Section 5 introduces the subspace hiding assumptions that we use to prove the security of our iaiO construction. Section 6 proves the security of our iaiO construction based on the subspace hiding assumptions presented in Section 5. The proofs of our subspace hiding assumptions from RHS are broken into several sub-arguments, described in Section 7, with some proofs of basic lemmata delegated to Appendix A. We also provide a diagram-based outline the overall structure and arrangement of our proofs in Figure 12 (Section 5). Section 8 discusses how to build iO from various relaxations of RKHwPRFs. Finally, Section 9 presents concluding remarks and discusses some open questions.

On the Length and Organization of the Paper. We acknowledge that our paper is very long; we are currently unable to compress our constructions and proofs into a form that would fit succinctly within a significantly smaller number of pages. This, unfortunately, seems unavoidable given the generic nature of our constructions and proof arguments. We have attempted to address this in two ways: first, we state our results and main theorems concisely in the overview section, while also providing some high-level intuition for our constructions and proofs. Next, in the main body, we partition our proofs into "bite-sized" arguments as best we can, and spend a substantial amount of text on proof overviews and intuition. We hope that this increases overall readability.

1.3 Discussion

Relationship between RKHwPRF and iO. In this paper, we show that RKHwPRFs imply iO. An interesting complimentary question is: could one could build RKHwPRFs from iO and other structured assumptions? We outline a possible approach towards answering this question:

- As a first step, one could hope to build RKHwPRFs from graded encodings with appropriate hardness assumptions (we make partial progress towards answering this question in this paper).
- The next step would be to show that iO and other structured assumptions together imply such graded encodings with the desired hardness assumptions.

As outlined earlier in our contributions, and discussed subsequently in Section 2.6 and Section 8, we show that a graded encoding with appropriate hardness assumptions (such as SXDH) over the source groups implies a slotted RKHwPRF, which can be viewed as an RKHwPRF with bounded multiplicative homomorphism (i.e., where the maximum number of homomorphic multiplication operations is predetermined). We note that analogous relaxations are frequently used in work involving lattice-based key-homomorphic PRFs such as [BLMR13], where the authors defined an almost-KHPRF that allowed for a predetermined number of homomorphic addition operations. In particular, our construction of iO from classic (unbounded) RKHwPRFs can be adapted to work with slotted RKHwPRFs, so long as the predetermined bound on the number of homomorphic multiplication operations is at least $(L + N - 1)$, 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. Achieving a full-fledged RKHwPRF from a graded encoding seems fundamentally hard since a graded encoding (as defined in [GGH13a]) inherently supports only a bounded number of bilinear map operations, and hence, only a bounded number of multiplications "in the exponent". This naturally translates to a construction of RKHwPRF with bounded multiplicative homomorphism.

We now turn to the question of building such graded encodings with appropriate hardness assumptions from iO in conjunction with additional structured assumptions. Note that certain prior works (such as [AFH+20] and [FHHL18]) have shown how to build graded encodings with various hardness assumptions from (probabilistic) iO, additively homomorphic encryption, NIZKs (with perfect soundness and perfect zero-knowledge) and a group where a strong variant of the DDH assumption holds. However, the resulting graded encodings are only equipped with hardness assumptions (concretely, the multilinear DDH assumption) over the final target group of the graded encoding. However, we require graded encodings with hardness assumptions over the source group to build slotted RKHwPRFs; in particular,

hardness assumptions over the target group do not suffice. Unfortunately, building a graded encoding with hardness assumptions over the source group from iO and structured assumptions seems to be an extremely difficult question that we leave as a very challenging open question.

Comparison with Rings with Inefficient Inversion. We present some high-level commentary comparing our framework for constructing iO from RKHwPRFs/RHS with the treatment of rings with inefficient inversion in [YYHK20]. We then discuss some of the challenges that could potentially arise when trying to generalize the framework of rings with inefficient inversion to make it similar to an RHS/RKHwPRF.

At a high level, a ring with inefficient inversion offers a similar syntax to that of an RHS (or RKHwPRF). However, there are a number of key differences. First of all, a ring with inefficient inversion is dependent on the ring being isomorphic to a finite field. In particular, since the ring scheme only gives out an encoding of the identity, it needs to be the case that all (or at least many) of the elements of the ring are reachable from additions or multiplications starting from the identity element (i.e., repeated doubling). This does not work at all for many rings (e.g., matrix rings). Secondly, the encoding of a ring with inefficient inversion behaves quite differently from an RHS. We need the PRF/synthesizer structure in order to obtain decision assumptions (which are necessary for iO). While something analogous to the c-inversion assumption from [YYHK20] would hold in an RHS, we need more for iO. We also note that a ring with inefficient inversion, due to requiring an isomorphism to a finite field, requires factoring to be hard (or probably some other equivalent assumption, as discussed in the paper). This means it most likely cannot be used for quantum-secure constructions. On the other hand, there is nothing about (non-exact or bounded) RHKwPRFs/RHS that make them inherently quantum insecure. Finally, due to the way it is defined, proofs for a ring with inefficient inversion must be done generically. All of our proofs and results for RHS/RKHwPRFs are done in the standard model.

In particular, we point out that an RHS/RKHwPRF does not imply a ring with inefficient inversion as defined precisely in [YYHK20]. This can be immediately seen from the fact that the output ring of an RHS does not have to be a prime-order ring of the form R_q for prime q, which is an explicit requirement in [YYHK20]. An RHS does, however, imply a ring where *some* associated "discrete log" problem is hard *in some form*. In particular, Theorem 3.11 describes the hardness assumption concretely. Note that this assumption would imply the hardness of more standard versions of discrete log over appropriately chosen rings (e.g., this assumption would imply the discrete log assumption over the ring R_q). If one were to consider such a relaxed interpretation of a ring with inefficient inversion, then an RHS does imply a ring with inefficient inversion.

To generalize the framework of rings with inefficient inversion to make it similar to an RHS/RKHwPRF, we would need the following:

- We would need to allow for arbitrary rings, or at least more general types of ring. While it is true that there are some rings where it is probably impossible to have any kind of reasonable hardness in our context (for instance, F_2^n), there could be many other interesting rings that might work. In particular, any secure RHS/RKHwPRF can be expanded into a matrix form, meaning that if a secure RHS/RKHwPRF exists, a matrix form of it (which would be incompatible with R_q or anything "close" to it) would also exist.
- We would also need to allow for encoding of arbitrary elements.
- We would need to have some kind of extra randomization input in the encoding process and allow for multiple calls to the ring generation algorithm RGen in [YYHK20] for similar parameters (if not already explicitly allowed).
- Finally, we would need to assume the RGen algorithm is weakly pseudorandom.

We point out that, even after making these large number of changes to the framework of [YYHK20], the resulting schemes would not be exactly equivalent. Hence, we think that rings with inefficient inversion, as described in [YYHK20], are considerably different from our treatment of RHS/RKHwPRF.

Alternative Approaches to Realize iO. Related to the above discussion, we note that the authors of [YYHK20] showed how to use rings with inefficient inversion to realize self-bilinear maps equipped with certain computational

assumptions that imply multiparty NIKE. It is natural to ask if an analogous approach could yield iO. Unfortunately, based on our current understanding of [YYHK20], we feel that this approach would not (directly) yield iO.

In particular, as per [YYHK20], the natural hardness assumptions over rings with inefficient inversion are only *computational assumptions*, and *not decision assumptions*. Hence, they can only be used to (directly) build self-bilinear maps with computational assumptions, which in turn only imply MMaps with computational assumptions. The only MMaps known to imply iO are ones where relatively strong decisional assumptions hold (e.g., multilinear SXDH). Building iO from MMaps with only computational assumptions would be a significant breakthrough, and is likely to require very involved techniques. In particular, we do not immediately see how to adopt rings with inefficient inversion as defined in [YYHK20] to support the kind of decision assumptions that we make over RHS/RKHwPRFs in this work. We leave exploring such extensions and resulting approaches towards constructing iO as very interesting future work.

Comparison with Multilinear Maps. A very natural question to ask is the following: how are RKHwPRFs (and their weaker variants) different from multilinear maps? On the surface, it might seem that RKHwPRFs are, in fact, functionally identical to multilinear maps. However, it turns out that there are substantial differences between multilinear maps and RKHwPRFs due to the weaker structure and simpler security definitions of RKHwPRFs.

While there have been several definitions of multilinear maps, the most widely accepted definitions of multilinear maps (identical to or based on those of [GGH13a]) require encoding *field* elements. This imposes quite a bit of extra structure compared to an RKHwPRF, which works over arbitrary rings. In fact, the authors of [AMP23] showed that field-homomorphic wPRFs cannot exist, so the difference between fields and rings can be substantial. Additionally, (most) multilinear map definitions require a user to be able to encode arbitrary elements and still have security. RKHwPRFs only have security for uniformly sampled input elements, which is a seemingly weaker requirement.

We additionally note that certain hardness assumptions over a multilinear map can be quite complicated and hard to describe concisely. RKHwPRFs, on the other hand, have a very simple security definition–that of a weak PRF game. To summarize, we feel that it is not straightforward to directly compare RKHwPRFs and multilinear maps in terms of their underlying hardness assumptions.

Relationship to Generic/Idealized Models. As pointed out in [AMP23], there is an important distinction between RKHwPRFs and ideal primitives like generic multilinear maps or graded encodings, and there are many advantages to constructions using standard-model primitives. In particular, constructions and separations in idealized models can be too powerful. For instance, as pointed out in [AMP23] as a concrete example, virtual black-box obfuscation is realizable in the generic graded encoding model [BR14, BGK+14], but not in the standard model [BGI+01].

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, we show that most of the well-known cryptographic primitives can seemingly be constructed in a generic manner from simple Minicrypt primitives endowed with additional algebraic structure (excluding a handful of advanced primitives that seem to require very special kinds of structure).

2 Technical Overview

In this section, we explain the intuition behind our iO construction and the corresponding proof of security at a high level. We refer to the remainder of the paper (where the formal definitions and proofs are located) as appropriate.

2.1 Definitions and Technical Tools

We begin by recalling (informal) definitions of a *ring key-homomorphic weak PRF* (RKHwPRF) and a *ring-embedded homomorphic synthesizer* (RHS) from [AMP23]. We also (informally) recall certain related technical tools from [AMP19] and [AMP23] that form the foundations for our construction and proof techniques. We refer the reader to Section 3 for the formal descriptions.

Ring Key-Homomorphic weak PRF (RKHwPRF). Based on the definitions introduced in [AMP23], 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 key space $(\mathcal{K}, \oplus, \otimes)$ and the output space $(\mathcal{Y}, \boxplus, \boxtimes)$ are rings with efficiently computable ring operations, the following hold:

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

Ring-embedded Homomorphic Synthesizer (RHS). The authors of [AMP23] showed that the above definitions can also be extended in the context of *pseudorandom synthesizers*. 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 uniformly 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.

The authors of [AMP23] 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

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

The definition of ring-embedded homomorphic synthesizer (RHS) in [AMP23] 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).

Key Homomorphism and Subset Sum. To understand our construction, it is important to understand the main intuition from [AMP19]. In this work, the authors use the property that random linear combinations using the same (secret) randomness for all the combinations *over the output space of a KHwPRF* are indistinguishable from random. To see this, suppose (following the example in [AMP23]) we 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:

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

Note that if $n > 3 \log |\mathcal{K}|$, then k' is distributed *statistically* close to random by the leftover hash lemma [IZ89]. In turn, this means that the entire bottom row is indistinguishable from random even given all of the other entries (except the $\mathbf{b}_i$s). Intuitively, this is like a knapsack problem, except that we also have the property that the final output is indistinguishable from random. Furthermore, since KHwPRF outputs are indistinguishable from random in the output space $\mathcal{Y}$, we can argue that similar sums of random elements in $\mathcal{Y}$ are distributed uniformly randomly as well.

Ring Homomorphism and Subset Sum. We now recall a technical tool from [AMP23] that we use in our construction of iaiO. 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

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

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, which was detailed in [AMP23]. 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

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

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. In [AMP23], the authors developed 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. We refer to Section 3.4 for a more detailed description of this observation and the associated proof techniques. Looking ahead, this observation allows us to prove a significantly more complicated, program-independent assumption over the output ring of any RKHwPRF (and, more generally, any RHS), that forms the basis of our construction of iO and the corresponding security proof.

2.2 Overview of Our Approach

We now present an informal overview of our main result and approach.

Branching Programs and Barrington's Theorem. Like many papers that build iO constructions, we will utilize branching programs and Barrington's theorem [Bar86]. Informally speaking (and according to Barrington himself), a branching program is just a "flowchart", composed of nodes, levels, and arrows between nodes. At each node, the program queries a single input bit and, based on that, moves to one of two follow-up nodes on the next level. For context, a *decision tree* is a branching program with fan-in one; generic branching programs are generally allowed arbitrary fan-in.

Typically, the nodes of branching programs are organized into "levels" of size w, with two requirements: first, the two possible follow-up nodes of a node on level i must be on level $i+1$. Second, all nodes on the same level query the same input variable. We call the maximum w over all levels in the branching program the *width* of the branching program. Barrington showed the following theorem about branching programs:

Theorem 2.1. ***Barrington's Theorem (Informal).*** *Let $f : \{0,1\}^n \to \{0,1\}$ be a function that is computable by a Boolean circuit of depth d. Then there exists a branching program of width 5 with $O\left(4^d\right)$ levels that computes f.*

Barrington additionally shows a similar result for the converse, proving equivalence of $\mathcal{NC}^1$ and efficiently computable branching programs of width 5, but we do not need this extra result here. Instead, we will need to examine some of the proof techniques used in Barrington's result.[1] In particular, the proof exploits the fact that branching programs of width 5 can be computed using permutations on 5 elements, or, in other words, the group S_5.

The main proof technique involves representing branching programs with n slots as a collection of $2n$ elements in S_5, such that the evaluations of the program on any n-bit input is given by a subset product of these $2n$ elements that selects one of the two elements corresponding to each slot. More concretely, For a branching program P represented by certain (pre-selected) elements $\{s_{i,b} \in S_5\}$ for $i \in [n]$ and $b \in \{0,1\}$, the evaluation of the program P on a given input $\mathbf{x} = (x_1, \dots, x_n)$ is given by the subset product $\prod_{i=1}^{n} s_{i,x_i}$, where x_i is the value of the input bit corresponding to the ith slot of the program (note that input bits can be potentially reused across multiple slots).

As elegantly pointed out in [Rao20], there are several core properties of cyclic permutations on 5 elements (alternatively, cyclic elements of S_5) that the proof of Barrington's theorem exploits. Recall that a cyclic permutation is one that, when iteratively applied to any element of the set, will reach all elements of the set before returning to its starting element. Cyclic permutations have the following nice properties:

[1] For a thorough explanation of Barrington's work, we recommend the Boppana-Sipser survey [BS89] or Anup Rao's course notes [Rao20].

- If $s_1, s_2 \in S_5$ are cyclic, then so is $s_1 s_2$.
- If $s \in S_5$ is cyclic, then so is s^{-1}.
- **Commutator Property**: there are two cyclic permutations $s_1, s_2 \in S_5$ with the additional property that $s_1 s_2 s_1^{-1} s_2^{-1}$ is another cyclic permutation.
- **Conjugation Property**: for any two cyclic permutations $s_1, s_2 \in S_5$, there is a permutation $t \in S_5$ (not necessarily cyclic) such that $t s_1 t^{-1} = s_2$.

It is straightforward to show how to use this logic to construct circuits recursively. Consider a circuit invariant that requires we output the identity permutation if the output of a function is 0, and a particular cyclic permutation s_O if the output is one. Suppose we have some circuit $g(x)$ and we want to compute $NOT(g(x))$. Multiplication by s_O^{-1} changes the output from $0 \to s_O^{-1}$ and $1 \to \mathbf{I}$, where $\mathbf{I}$ is the identity permutation. To preserve the circuit invariant, recall that, by the conjugacy property, there exists some t such that $t s_O^{-1} t^{-1} = s_O$. By definition, we also have $t\mathbf{I}t^{-1} = \mathbf{I}$, so therefore we can apply a NOT gate to $g(x)$ as $t g(x) s_O^{-1} t^{-1}$.

AND gates are slightly more complicated, but still explainable quickly. Consider two permutations $s_1, s_2 \in S_5$ that satisfy the commutator property above. Note that $s_1 s_2 s_1^{-1} s_2^{-1}$ is equal to some cyclic permutation, but $s_1 \mathbf{I} s_1^{-1} \mathbf{I}^{-1}$, $\mathbf{I} s_2 \mathbf{I}^{-1} s_2^{-1}$, and, of course, $\mathbf{III}^{-1}\mathbf{I}^{-1}$ are all equal to the identity permutation. Given two input programs $g(x)$ and $h(x)$ that we want to AND together, we can use the conjugation property to put them in the right initial state, apply the above logic, and then apply the conjugation property again to get the desired output element in the case the output is 1. We can use these techniques to construct any gates we like.

Finally, we note that, in the rest of this paper, we will represent elements in S_5 as permutation matrices of dimension 5. This is a very straightforward mapping; for instance, the core logic of an AND gate of the form $s_1 s_2 s_1^{-1} s_2^{-1}$ might look something like the following:

$$\begin{bmatrix} & & & & 1 \\ 1 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & 1 & \end{bmatrix} \begin{bmatrix} & 1 & & & \\ & & & 1 & \\ 1 & & & & \\ & & & & 1 \\ & & 1 & & \end{bmatrix} \begin{bmatrix} & 1 & & & \\ & & 1 & & \\ & & & 1 & \\ & & & & 1 \\ 1 & & & & \end{bmatrix} \begin{bmatrix} & & 1 & & \\ 1 & & & & \\ & & & & 1 \\ & 1 & & & \\ & & & 1 & \end{bmatrix}$$

$$= \begin{bmatrix} & & & 1 & \\ & & 1 & & \\ 1 & & & & \\ & & & & 1 \\ & 1 & & & \end{bmatrix}$$

Looking ahead, this will be the form in which we use programs in our construction.

The Starting Point: Input-Activated iO. Rather than trying to prove iO "from scratch", we use the framework of [GLSW15] in order to prove iO security, as this is a very clean way to build iO from a program-independent assumption and in the standard model. More precisely, we build an *input-activated* iO (iaiO) scheme and then simply adopt the machinery from [GLSW15] which shows that an iaiO scheme implies a full iO scheme.

At its core, iaiO is a way of starting with an initial set of programs and then adding modifications so that the program can be "turned off" on an arbitrary set of inputs. Unfortunately, iaiO is considerably more complicated than just this, but its complexity makes it considerably easier to prove from a simple assumption than regular iO (and this is presumably by design in [GLSW15]).

More precisely, iaiO considers a set of ℓ programs $P_1, ..., P_\ell$ and an array over $n \times \ell \times 2$ with binary elements, which, following [GLSW15], we will typically view as an $n \times \ell$ matrix $\mathbf{G}$ with both a 0 and a 1 "slot" in each matrix entry. We will refer to entries in this matrix in the form $\mathbf{G}_{i,j,b}$.

For each column $j \in \{1, ..., \ell\}$ of the matrix, we define a function $f_j : \{0,1\}^n \to \{0,1\}$ that describes on which inputs the jth column (or, equivalently, the program P_j) is active. On an input $\mathbf{x} \in \{0,1\}^n$, $f_j(\mathbf{x}) = 1$ if and only if $\mathbf{G}_{i,j,\mathbf{x}_i} = 1$ for all $i \in \{1, ..., n\}$. This might seem an unusual way to activate inputs, but, looking forward, we

will work with matrix representations of branching programs (as we can represent all programs in $\mathcal{NC}^1$ this way), so this will let us directly map to turning "on" and "off" inputs because they correspond nicely to matrices for matrix representations of a program P_j.

Correctness of iaiO. Correctness for iaiO is also defined in terms of activated inputs. For each input $\mathbf{x} \in \{0,1\}^n$, we define a set $S_{\mathbf{x}} \subseteq \{1, ..., \ell\}$ that includes all j for which $f_j(\mathbf{x}) = 1$. In other words, $S_{\mathbf{x}}$ contains all of the indices corresponding to the programs that are activated on input $\mathbf{x}$. Correctness requires that, for all inputs $\mathbf{x}$, if $S_{\mathbf{x}} \neq \emptyset$ and $P_j = P_{j'}$ for all $j, j' \in S_{\mathbf{x}}$, then the evaluation of an obfuscated program T on input $\mathbf{x}$ must be equal to $P_j(\mathbf{x})$ for all $j \in S_{\mathbf{x}}$.

Note that this heavily restricts the cases where iaiO is actually required to be correct. If the programs are not activated for particular inputs, the iaiO is not required to be correct on those inputs. Furthermore, correctness must hold only for programs with *exactly identical* program descriptions, not just those with identical input/output behavior.

Security of iaiO. Unlike traditional iO, security for iaiO is considerably more difficult to define. There are four security games that we need to consider, which we explain below. In all of our games, we assume our $n \times \ell \, (\times 2)$ matrix $\mathbf{G}$ as well as a list of programs $P_1, ..., P_\ell$. We will defer the security parameter to the full definition.

- **Inter-column Security Game.** In this game, we consider two columns denoted by j and k, as well as some row index i^*. We also consider a slot index $\beta \in \{0,1\}$ such that $\mathbf{G}_{i^*,j,\beta} = 1$. For every row i and slot $\gamma \in \{0,1\}$, except for $i = i^*$ and $\gamma = 1 - \beta$, we require that if $\mathbf{G}_{i,k,\gamma} = 1$, then $\mathbf{G}_{i,j,\gamma} = 1$ too. Importantly, we also require that $P_j = P_k$ (i.e. the programs are exactly identical).

  Informally speaking, this game is to distinguish whether or not an iaiO with this structure has $\mathbf{G}_{i^*,k,\beta} = 0$ or 1. An important thing to note about this game is that changing $\mathbf{G}_{i^*,k,\beta}$ does not change the overall set of inputs that are active, because any input that could be activated (or deactivated) by changing $\mathbf{G}_{i^*,k,\beta}$ is going to still be active in the jth column (i.e. P_j will still evaluate that input) and thus active in the overall iaiO.

- **Intra-column Security Game.** In this next game, we consider some column j of $\mathbf{G}$ and some particular row i^* where both slots are equal to 0. Then, suppose we consider some other arbitrary column C (potentially not present in $\mathbf{G}$) such that the row i^* also has both slots equal to 0.

  This game involves distinguishing whether an iaiO was done faithfully (using column j) or whether C was used in the iaiO in place of j. Note that, since both the "real" column j and "fake" column C have a row i^* with both slots 0, it will be the case that $f_j(\mathbf{x}) = f_C(\mathbf{x}) = 0$ for all inputs $\mathbf{x}$, meaning that this change will not change the inputs activated overall for the iaiO.

- **Completely Inactive Program Security Game.** We now consider a column j where all entries of the form $\mathbf{G}_{i,j,\beta}$ are 0 for all i and β. This game is to decide whether or not the challenger has replaced the initial program P_j in column j with a different program P^*. Note that this also does not change the behavior of the overall iaiO since all inputs are deactivated.

- **Single-input Program Switching Security Game.** Consider a column j where the function f_j is equivalent to a point function (so $\mathbf{G}_{i,j,\beta}$ has exactly one $\beta = 1$ for every i) on a point x^*. In addition, consider a program P^* such that $P_j(x^*) = P^*(x^*)$. The game is to distinguish whether or not the iaiO was performed honestly with P_j or with P^* substituted instead. Again, the behavior of the overall iaiO is unchanged because only the single input x^* is active on this column.

Connection between Branching Programs and iaiO. It can be difficult to conceptualize the connection between branching programs and iaiO, so we present a simple example here. We will consider a two-input, three-program instance that computes an AND gate (on the two bits of input), and ignore many extra additions to the construction that are essential for security. Recall that earlier we showed an AND gate using a Barrington program might look like $p_1 p_2 p_1^{-1} p_2^{-1}$, where p_1 and p_2 are either the permutation matrices below (our old matrices s_1 and s_2, if the input bits

are one) or the identity matrix (if the input bits are zero):[1]

$$\begin{bmatrix} & & & & 1 \\ 1 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & 1 & \end{bmatrix} \begin{bmatrix} & 1 & & & \\ & & & 1 & \\ 1 & & & & \\ & & & & 1 \\ & & 1 & & \end{bmatrix} \begin{bmatrix} & 1 & & & \\ & & 1 & & \\ & & & 1 & \\ & & & & 1 \\ 1 & & & & \end{bmatrix} \begin{bmatrix} & & 1 & & \\ 1 & & & & \\ & & & & 1 \\ & 1 & & & \\ & & & 1 & \end{bmatrix}$$

$$= \begin{bmatrix} & & & 1 & \\ & & 1 & & \\ 1 & & & & \\ & & & & 1 \\ & 1 & & & \end{bmatrix}$$

We can expand this into a full program by choosing vectors $\mathbf{t}_0 := \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \end{bmatrix}$ and $\mathbf{t}_1^T := \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \end{bmatrix}$. We emphasize that we could have chosen different vectors here, as we just need to check if a single bit in a matrix corresponding to one of the non-diagonal bits of the above permutation matrix is nonzero. Then our program can be just $t_0 p_1 p_2 p_1^{-1} p_2^{-1} t_1$. It is straightforward to verify that this successfully computes the AND function. But what about an entire iaiO program with multiple programs being computed in parallel? Consider the following expansion:

$$\begin{bmatrix} \mathbf{t}_0 & \mathbf{t}_0 & \mathbf{t}_0 \end{bmatrix} \begin{bmatrix} p_1 & & \\ & p_1 & \\ & & p_1 \end{bmatrix} \begin{bmatrix} p_2 & & \\ & p_2 & \\ & & p_2 \end{bmatrix} \cdot$$

$$\begin{bmatrix} p_1^{-1} & & \\ & p_1^{-1} & \\ & & p_1^{-1} \end{bmatrix} \begin{bmatrix} p_2^{-1} & & \\ & p_2^{-1} & \\ & & p_2^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{t}_1 \\ \mathbf{t}_1 \\ \mathbf{t}_1 \end{bmatrix}$$

This program evaluates to 3 if the AND is true and 0 if it is false, and is just three instances of the same program running in parallel, with the output summed up. We can make things a little bit cleaner by making the nonzero entries of our permutation matrix random elements in some large finite field $\mathbb{F}_q$, and in this case the output will be zero if the AND is false, and, with all but negligible probability, nonzero if the AND is true. For now, we ignore this due to the extra complexity but, looking ahead, our constructions will use these kinds of randomization techniques.

Again, we can view the above program as really three programs running in parallel, P_1, P_2, and P_3, with program i corresponding to the nonzero slots on row/column i in the matrices above. Now how can we turn this into an iaiO prototype? The answer is pretty straightforward: if the activation matrix $\mathbf{G}$ has a zero in a certain slot, we replace the (sub)matrices in our program corresponding to the (sub)matrices representing those input bits with the all zeroes matrix. To see this, consider the following activation matrix:

$$\mathbf{G} = \begin{bmatrix} 1/1 & 1/0 & 1/1 \\ 1/1 & 1/0 & 0/1 \end{bmatrix}$$

Recall that each column j of the activation matrix determines which program P_j is active. In addition, recall that on an input $\mathbf{x} \in \{0,1\}^n$, the activation function $f_j(\mathbf{x}) = 1$ if and only if $\mathbf{G}_{i,j,\mathbf{x}_i} = 1$ for all $i \in \{1, ..., n\}$. Then, on input $\mathbf{x} = 01$, P_1 is clearly active because all of the bits in the first column are 1. P_2 is inactive because $\mathbf{x}_2 = 1$ and $\mathbf{G}_{2,2,1} = 0$. P_3 would be inactive on input 10 or 00 because $\mathbf{G}_{2,3,0} = 0$, but is active because the second bit of $\mathbf{x}$ is one.

As we alluded earlier, we use the identity matrix for zero inputs and specific permutation matrices for one inputs for the AND gate, so, in our construction, this would lead to something similar to the following evaluation:

$$\begin{bmatrix} \mathbf{t}_0 & \mathbf{t}_0 & \mathbf{t}_0 \end{bmatrix} \begin{bmatrix} \mathbf{I} & & \\ & \mathbf{I} & \\ & & \mathbf{I} \end{bmatrix} \begin{bmatrix} s_2 & & \\ & 0 & \\ & & s_2 \end{bmatrix} \cdot$$

[1] The actual gate would be more complicated in a more complicated circuit, but we'll ignore that for now.

$$\begin{bmatrix} \mathbf{I} & & \\ & \mathbf{I} & \\ & & \mathbf{I} \end{bmatrix} \begin{bmatrix} s_2^{-2} & & \\ & 0 & \\ & & s_2^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{t}_1 \\ \mathbf{t}_1 \\ \mathbf{t}_1 \end{bmatrix}$$

Note that the above expression evaluates to $2\mathbf{t}_0\mathbf{I}\mathbf{t}_1 = 0$, as desired. An astute reader might ask the following question: why are the "middle" identity matrices in the above expression present (and not zero)? The answer is because even though the second program P_2 isn't active on our choice of $\mathbf{x}$, it is the case that $\mathbf{G}_{1,2,0} = 1$ and it could be active on input $\mathbf{x} = 00$. Looking ahead, leaving these extra terms in place even though we could eliminate them and get the same output will make our hybrid arguments considerably easier.

This nice mapping between iaiO and branching programs was utilized in [GLSW15], where their construction worked by essentially evaluating branching programs in a form similar to the one above "in the exponent" of a multilinear map, although there is a lot of extra complexity required for security that we omit right now (but will get to soon). Our construction will use (less structured) RKHwPRFs instead, as we talk about next.

(ia)iO from Generic RKHwPRF/RHS. Our main contribution is a construction of iO for $\mathcal{NC}^1$ from any generic RHS (and hence, from any generic RKHwPRF). We will generally explain our techniques here in terms RKHwPRFs in order to simplify the presentation, although they also follow from any RHS.

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 special composite-order 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.[1] 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.

Our Approach. We construct input-activated iO from generic RKHwPRFs as follows: we introduce an assumption 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 generic RKHwPRF by building upon the technical tools presented (informally) in Section 2.1. We then show that such an assumption implies an input-activated iO scheme. Note that we do not explicitly use the RKHwPRF primitive in the construction; we simply use the ring R which is the output space of the RKHwPRF (along with the associated hardness assumption over R as mentioned earlier). Informally, we prove the following theorem:

Theorem 2.2. (Informal) *Assuming the existence of any RKHwPRF (and more generally, any ring-embedded homomorphic synthesizer), we can construct a secure input-activated iO scheme as defined in [GLSW15] for all circuits in* $\mathcal{NC}^1$.

2.3 A Simplified Version of Our iaiO Construction

We now present a simplified overview of our approach for designing an input-activated iO scheme from any RHS. Using [GLSW15], we can leverage this into a full iO scheme for all functions in $\mathcal{NC}^1$. We present all of the relevant definitions and notions of security in Section 3, while the detailed construction and proof of security are presented in Sections 4 and Section 5, as well as across Sections 6 and 7.

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 RKHwPRF. We refer the reader to Section 4 for the details of the embedding technique.

[1] 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.

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 [BR14] 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 [BR14] 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.

Below, we describe a simplified version of our construction that captures many of the ideas that the full construction builds on.

Permutation Branching Programs. Recall that we can represent any $\mathcal{NC}^1$ program as an oblivious permutation branching program P of width 5. Once again, we set $P = \{M_{\ell,0}, M_{\ell,1}\}_{\ell\in[L]}$, where L denotes the depth of P and each $M_{\ell,b} \in \{0,1\}^{5\times 5}$ is a permutation matrix. In addition to what we have discussed before, we 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 ℓth level branch.

Kilian-Style Randomization. We uniformly sample 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.

Ring-Embedding Permutation Matrices. We will now use the following strategy to embed a permutation matrix into the output ring R of a generic RKHwPRF. Given a permutation matrix $N_{\ell,b} \in \{0,1\}^{5\times 5}$, 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:

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

Generating Guard Matrices. We generate a sequence of "guard" matrix-pairs

$$\{(\mathbf{L}_\ell, \mathbf{R}_\ell)\}_{\ell\in[L]}$$

over the ring R such that the following conditions hold:

- For each $\ell \in [L]$, the "left guard" $\mathbf{L}_\ell$ is of dimension $c_1 m \times c_2 m$ and the "right guard" $\mathbf{R}_\ell$ is of dimension $c_2 m \times c_1 m$ where c_1, c_2 are parameters such that $c_1 \gg c_2$. 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

$$\mathbf{D}_\ell = \mathsf{diag}(\mathbf{D}_{\ell,1}, \mathbf{D}_{\ell,2}, \ldots, \mathbf{D}_{\ell,c_2 m}),$$

  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 refer the reader to Section 4 for the details of how such a sequence of "guard" matrix-pairs can be generated efficiently. 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 $R^{m \times m}$.

The (Simplified) Construction. We are now ready to describe a simplified version of our iaiO 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 parameters $c_2 = 5$ and $c_1 \gg 5$.

3. **Step 3:** Generate an obfuscated program P consisting of a sequence of $2L$ "guarded" program matrix encodings

$$\mathsf{P} = \{\widetilde{\mathbf{N}}_{\ell,b}\}_{\ell \in [L], b \in \{0,1\}},$$

   where $\widetilde{\mathbf{N}}_{\ell,b} = \mathbf{L}_\ell \mathbf{N}_{\ell,b} \mathbf{R}_\ell$.

(Simplified) Evaluation. 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 a 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 $Q = \prod_{\ell=1}^{L} M_{\ell, x_{\phi(\ell)}}$. The evaluation procedure crucially uses the following relation that holds with overwhelmingly large probability between each sub-matrix $\mathbf{Q}_{w,v}$ 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 evaluator will now perform a *zero test* as follows: choose a particular non-diagonal entry (i, j) such 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 [1]. It suffices to check whether the corresponding sub-matrix $\mathbf{Q}_{i,j}$ 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.

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

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

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

$$\mathbf{C}_{(\ell,\ell'),(b,b')} = \begin{cases} \mathbf{0} & \text{if } (\ell', b') = (\ell, 1 - b), \\ \text{uniform in } R^{5m \times 5m} & \text{otherwise.} \end{cases}$$

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 $5m(2L + 1) \times 5m(2L + 1)$, structured as:

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

where for each $i \in [N]$ and each $b \in \{0, 1\}$, $\mathbf{T}_{i,b}$ is a uniform *block-diagonal* matrix in $R^{5m \times 5m}$, while for each $i \in [N]$, each $\ell \in [L]$, and each $b, b' \in \{0, 1\}$, we have

$$\mathbf{C}_{(i,\ell),(b,b')} = \begin{cases} \mathbf{0} & \text{if } (i, b') = (\phi(\ell), b), \\ \text{uniform in } R^{5m \times 5m} & \text{otherwise,} \end{cases}$$

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 ℓth level branch.

[1] As we explain in Section 2.2 when discussing Barrington's theorem (Theorem 2.1), the final output matrix Q is either the identity matrix or a "particular" permutation matrix that is different from the identity matrix, so we know exactly which single coordinates are viable ones to check when evaluating the obfuscated program.

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 $c_1 \gg 2(L+N)+1$. Finally, we output an obfuscation P consisting of a sequence of $(2L+2N)$ "guarded" matrix encodings of the form:

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

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, 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.

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

$$\mathbf{Q} = \left(\prod_{\ell=1}^{L} \widetilde{\mathbf{P}}_{\ell, x_{\phi(\ell)}} \right) \left(\prod_{i=1}^{N} \widetilde{\mathbf{E}}_{L+i, x_i} \right).$$

Note that because of the structuring of the end-guard matrices, $\mathbf{Q}$ is a $5m \times 5m$ matrix of elements in the ring R.

As in the simple construction described earlier, the zero test will choose a particular non-diagonal matrix position (i,j) so that the plaintext entry $Q_{i,j}$ is non-zero in the subset-product matrix Q whenever the branching program outputs 0 (i.e. whenever the matrix product Q is not the identity permutation matrix). At this point, the zero test simply checks if $\mathbf{Q}_{i,j} = \mathbf{0}$. 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 $(\ell, b) \in [L] \times \{0,1\}$, if $b \neq x_\phi(\ell)$, then the consistency sub-matrices in the corresponding diagonal slot do not contribute to the final product since there is a zero sub-matrix in this diagonal slot in the program carrying matrix $\mathbf{P}_{\ell, x_{\phi(\ell)}}$.
2. For each $(\ell, b) \in [L] \times \{0,1\}$, if $b = x_\phi(\ell)$, then the consistency sub-matrices in the corresponding diagonal slot do not contribute to the final product since there is a zero sub-matrix 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 $\mathbf{Q}_{i,j}$ come from the diagonal slots corresponding to the ring-embedded permutation matrices. It is now easy to see that $\mathbf{Q}_{i,j} \neq \mathbf{0}$ 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 4 for details on how to handle input activations for multiple programs being executed in parallel, and to Section 5 and Sections 6 and 7 for the detailed proof of security.

2.4 Security Proof Intuition

We next describe some intuition as to why our construction of iaiO is secure, assuming we are given a secure RHS (or a secure RKHwPRF, which implies a secure RHS). In the full proof, we explain considerably more intuition (including why we need the guard matrices, enforcer matrices, etc., and how they all work). Recall that, fundamentally, the

challenge of showing iaiO security is "erasing" or "adding" matrices to our (guarded) Barrington programs. Recall that our guard matrices have the form $\mathbf{L}_{i,b}$ and $\mathbf{R}_{i,b}$, and suppose matrices $\mathbf{X}_{j,i}$ are matrices corresponding to nonzero entries in the Barrington program. What we really need to do in our proof is show things like the following are indistinguishable:

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

and

$$\begin{bmatrix} | & | \\ \mathbf{L}_{2,1} & \mathbf{L}_{2,2} \\ | & | \end{bmatrix} \begin{bmatrix} \mathbf{X}_{1,i} & \mathbf{0}_R \\ \mathbf{0}_R & \mathbf{0}_R \end{bmatrix} \begin{bmatrix} -- & \mathbf{R}_{2,1} & -- \\ -- & \mathbf{R}_{2,2} & -- \end{bmatrix}.$$

The above instance is essentially a "snippet" of our full iaO program, where we are deactivating on the input corresponding to $\mathbf{X}_{2,i}$ on the program represented in the 2nd row/column in some hybrid. First, note that if all of these matrices were random, then the security proof would trivially follow from the assumption that subset sums over the RKHwPRF output space are indistinguishable from random, even with a repeated secret subset (what we explained in the very beginning of the section).

However, while the $\mathbf{X}$ matrices are (essentially) random "enough" for this to work, the guard matrices $\mathbf{L}$ and $\mathbf{R}$ are not. We cannot directly apply the RKHwPRF output space assumption because these matrices are correlated with other guard matrices! In addition, due to the way we need to prove iaiO security, we need to give out many samples of this general form. At first, it seems difficult how this proof might work.

There is some good news: in the indistinguishability proofs we need to do for iaiO, there will always be at least one "free" guard matrix that we can randomize. This is due to the clever way that the authors of [GLSW15] defined iaiO and this kind of structure–with a "free" subspace, if you will–is also necessary for their proof of iaiO from the multilinear subgroup elimination assumption to go through. That means we will have challenge terms of the form

$$\begin{bmatrix} | & | \\ \mathbf{L}_{2,1} & \mathbf{Z} \\ | & | \end{bmatrix} \begin{bmatrix} \mathbf{X}_{1,i} & \mathbf{0}_R \\ \mathbf{0}_R & \mathbf{X}_{2,i} \end{bmatrix} \begin{bmatrix} -- & \mathbf{R}_{2,1} & -- \\ -- & \mathbf{R}_{2,2} & -- \end{bmatrix}$$

where the matrix $\mathbf{Z}$ is uniformly random. It turns out that, with some work, we can show that this kind of expression is indistinguishable from expressions of the form

$$\begin{bmatrix} | & | \\ \mathbf{L}_{2,1} & \mathbf{Z} \\ | & | \end{bmatrix} \begin{bmatrix} \mathbf{X}_{1,i} & \mathbf{0}_R \\ \mathbf{0}_R & \mathbf{X}_{2,i} \end{bmatrix} \begin{bmatrix} -- & \mathbf{Z}' & -- \\ -- & \mathbf{R}_{2,2} & -- \end{bmatrix}$$

for a random $\mathbf{Z}'$, even with (potentially) some other guards present that are not presented here. While this is hopefully believable even at this point, the proof is slightly complicated and involves using a "trapdoor" for the guard matrices (i.e. setting $\mathbf{R} = \mathbf{R}_{half} || (\mathbf{R}_{half}\mathbf{A})$ for some square $\mathbf{A}$). The intuition for how this works should be straightforward for a reader familiar with lattice trapdoors, but our construction is significantly more complicated here because lattice trapdoors assume efficient inversion of the matrix elements (which are usually field elements), and we are stuck working with non-invertible rings.

These tools for rings are some of the core theoretical contributions we make in the paper and may be useful in other applications of cryptography that work over large rings.

Finishing up the Proof. With our basic indistinguishability lemmas that we've outlined above in mind, the rest of the proof is relatively straightforward, although quite long and tedious. The first step is to use hybrid arguments to "propagate" the random guard matrices across an entire two-program iaiO mini-instance. This is a tedious but mostly straightforward sequence of applying (essentially) the same lemma over and over until we completely transform a program.

Finally, the last step is to "lift" these mini-instances up into full iaiO instances. This is also relatively straightforward but quite tedious, and involves the "trapdoor" construction we alluded to earlier. The proof is quite long (unfortunately, seemingly by necessity), but once the basic indistinguishability arguments are well-understood, it should be smooth sailing for any reader.

Other Parts of the Proof. We haven't outlined here how we enforce input consistency or other properties that we need for a secure iaiO (or almost any iO) construction. However, these arguments are generally much simpler to make than what we've presented above, so we defer them to the formal proofs.

2.5 A Sufficient Ring Assumption for Our iaiO Construction

At this point, an astute reader may note that we have seemingly not used the full functionality of an RHS in our construction. Like previous work on (group) KHwPRFs [AMP19], our construction itself only uses the output space of the RHS and never explicitly needs to compute the RHS itself. The only time that the RHS computation ever needs to take place is in the security proof. A natural question to ask, therefore, is the following: why not directly characterize the requirements on the ring we need for our construction itself rather than use an RHS/RKHwPRF? In this section, we discuss a possible characterization, and then highlight some nuanced considerations that need to be taken into account.

2.5.1 Rings with a "Both-sided" Indistinguishability Assumption

Recall that, at the beginning of the overview, we showed a core property of the output ring of the RHS that we exploited: namely, for an output space $(\mathcal{Y}, \boxplus, \boxtimes)$ of an RHS having efficiently computable ring operations, if we *uniformly and independently* sample matrices $\mathbf{K}_x, \mathbf{S}_x^{(1)}, \ldots, \mathbf{S}_x^{(m)} \in \mathcal{Y}^{n \times n}$, then the tuple

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

is indistinguishable from random. We emphasize (and formally prove in Section 3.4) that this property holds for the output ring of *any* generic RHS (and, hence *any* generic RKHwPRF). In fact, we prove a *both-sided* version of the above property in Section 3.4, which states that for the output space $(\mathcal{Y}, \boxplus, \boxtimes)$ of *any* generic RHS having efficiently computable ring operations, if we *uniformly and independently* sample matrices

$$\mathbf{K}_x, \mathbf{K}'_x, \mathbf{S}_x^{(1)}, \ldots, \mathbf{S}_x^{(m)}, \mathbf{T}_x^{(1)}, \ldots, \mathbf{T}_x^{(m)} \in \mathcal{Y}^{n \times n},$$

then the tuples

$$\left(\begin{bmatrix} \mathbf{S}_x^{(1)} \\ \mathbf{S}_x^{(2)} \\ \vdots \\ \mathbf{S}_x^{(m)} \end{bmatrix}, \begin{bmatrix} \mathbf{S}_x^{(1)} \\ \mathbf{S}_x^{(2)} \\ \vdots \\ \mathbf{S}_x^{(m)} \end{bmatrix} [\mathbf{K}_x] \right), \quad \left(\begin{bmatrix} \mathbf{T}_x^{(1)} \\ \mathbf{T}_x^{(2)} \\ \vdots \\ \mathbf{T}_x^{(m)} \end{bmatrix}, [\mathbf{K}'_x] \begin{bmatrix} \mathbf{T}_x^{(1)} \\ \mathbf{T}_x^{(2)} \\ \vdots \\ \mathbf{T}_x^{(m)} \end{bmatrix} \right),$$

are *both* indistinguishable from random. This seemingly gives us a very simple, practical characterization of a ring that would suffice for our constructions: we just need a ring where the above both-sided version of the indistinguishability property holds. Looking ahead, in the main body of the paper, we indeed present the above characterization of a ring in terms of the double-sided indistinguishability assumption as the sufficient requirement for our iaiO construction (see Theorem 5.6), while additionally capturing the realization of such a ring based on any RHS (resp., any RKHwPRF) in Theorem 5.7 (resp., Corollary 3.15).

2.5.2 Additional Considerations

While the above seems like a very simple characterization of a ring that suffices to build iaiO, we point out some additional considerations that need to be taken into account. For example, we (implicitly) rely on efficient algorithms for sampling ring elements, as well as for addition, multiplication, and zero-testing of ring elements. These will probably not be natural ring operations for any real construction, and, worse, the output ring $\mathcal{Y}$ will probably be indistinguishable from random in some larger ring $\mathcal{Y}'$, and thus will require a specific sampling algorithm and parameters. To sample an

element of $\mathcal{Y}$ appropriately, we will probably need some public parameters x and also private randomness k, making the format very similar to a PRF. Even if we don't need the public parameters x, we still need to sample private randomness k, making our construction very similar to a synthesizer.

FHE / RIHwPRF Analogy. A useful consideration here is the result from [AMPR19] that shows that every FHE construction implies an RIHwPRF, and vice versa (so there is a full equivalence). When constructing an IHwPRF from an FHE scheme, the input space of the PRF is the space of valid FHE ciphertexts under a certain key, and the output space is the decryption of those ciphertexts. But, for all known FHE schemes, the ciphertext space for a particular key (corresponding to the input space for an IHwPRF) is very sparse in some larger space, and is hard to distinguish FHE ciphertexts from random in this larger space and sample without the public key for the FHE scheme. In fact, if there exists an FHE scheme where the ciphertext space is *independent* of the key[1], we actually show that this implies an RHS (and thus iO). The construction of such an FHE scheme would therefore obviously be a huge breakthrough.

Thinking back to our construction, in any practical construction of a "ring" that implies iaiO using our construction, it seems that we are likely to need potentially public (corresponding to an input x) and most likely private (corresponding to a key k) randomness in order to sample ring elements. This probably means that we either need the full PRF input/output behavior or at least the synthesizer behavior in the sampling algorithm of a description of any kind of ring that would work for our construction. Moreover, this sampling algorithm would (by definition) be a PRF (or synthesizer) itself. So, we believe that it is perfectly natural to define the requirements of our iaiO construction in terms of RHS/RKHwPRFs (as opposed to directly characterizing the ring in terms of the double-sided indistinguishability assumption outlined above), since these primitives have simple, concise definitions that are perhaps more immediately understandable by readers.

2.5.3 Future Work on Alternate Ring Characterizations

We believe that thinking more directly about the kind of rings needed for our construction to work would undoubtedly make for excellent future work. We show in this paper that any ring that is the output space of an RHS/RKHwPRF can be used to build iO, but we do not specify any additional properties of these rings (other than the obvious, double-sided indistinguishability property described above). Figuring out the properties of rings that do (or do not) work in our construction could potentially lead to new iO constructions (or impossibility results), and, in particular, potentially post-quantum constructions, since our framework (i.e., a generic RHS) is not inherently vulnerable to any quantum attack.

2.6 Relaxations of Classic RKHwPRFs

Many of the concrete primitives (e.g., graded encodings/multilinear maps) from which we wish to construct iO do not quite fit into the framework of a classic RKHwPRF. To accommodate this, we consider relaxed versions of "classic" RKHwPRFs that support all of the constructions from classic RKHwPRFs in essentially the same way. Additionally, these relaxations yield constructions of iO from simpler computational assumptions over graded encodings/multilinear maps. We present an overview of these definitions here.

Note that a classic RKHwPRF supports unbounded many addition and multiplication operations over its output ring. Our relaxations correspond to "bounding" the number of such operations that can be performed. Analogous relaxations are frequently used in work involving lattice-based KHPRFs such as [BLMR13], where the authors defined an "almost"-KHPRF that allowed for a pre-determined number of homomorphic operations.

Slotted RKHwPRFs. We present an overview of our first relaxation of classic RKHwPRFs, namely "slotted" RKHwPRFs. At a high level, we restrict the number of multiplication operations allowed over the output space of an RKHwPRF. We motivate this by showing how an asymmetric bilinear map, where the SXDH assumption holds, gives rise to a "two-slotted" RKHwPRF.

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

[1] This actually needs some extra conditions, which we explain later.

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). At this point, 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:

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

It is easy to see that $F_1(k_1, x) \odot F_2(k_2, x) = F_T(k_1 \cdot k_2, x)$. We refer to the ensemble $\{F_1, F_2, F_T\}$ as a "two-slotted"-RKHwPRF.

Clearly, the aforementioned construction does not formally meet the definition of a "classic" RKHwPRF. However, it seemingly provides similar algebraic structure. It supports unbounded addition operations and a single multiplication operation, with the added restriction that a multiplication is only allowed between an element from $\mathbb{G}_1$ (which constitutes "slot-1" of the output space) and an element from $\mathbb{G}_2$ (which constitutes "slot-2" of the output space). No other multiplications are allowed.

More generally, an n-slotted RKHwPRF resembles a collection of n parallel-secure weak PRFs that are divided into n slots. It closely resembles a classic RKHwPRF, modulo 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 classic 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. The number of allowed additive operations is 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. It turns that our construction of iO 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 construction 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 8.1, 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, for the first time, iO from any asymmetric multilinear map where SXDH holds.

Slotted Partial RKHwPRFs. In Section 8.2, 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 8.2, 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[2].

2.7 iO 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.

In Section 8.2, we prove that under the assumption that SXDH is hard over the MZ18 MMap encodings, a slotted partial RKHwPRF exists. 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 adapt 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 zeroizing 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.

2.7.1 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.

[1] Note that evaluating a classic/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.

[2] Once again, our constructions would not work for symmetric multilinear maps that only allow "one-shot" pairings of all n elements together to a final element in the target group.

The construction maps plaintext elements onto *encodings*, where each encoding is associated with a particular level. An MZ18 MMap of degree $N = \text{poly}(\lambda)$ (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 $(a \cdot b)$.

High-Level Intuition. The MZ18 map is an interval-based construction: it supports addition of identical intervals, multiplication of adjacent intervals, and a zero-test. These features almost immediately provide the functionality we need for building a partial slotted RKHwPRF; the tricky part is arguing that we can encode elements in a way that provides (weak) pseudorandomness.

On the surface, this might seem very simple. But there are some difficulties. For instance, in most multilinear maps it is possible to multiply elements by scalars. However, that is not possible with MZ18, so we have to generalize our techniques to handle situations that we could typically solve with scalar multiplication. We elaborate on this more later in this section.

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 ith 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 ith 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 ℓth 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 $\mathbf{R}_0, \mathbf{R}_1, \ldots, \mathbf{R}_{NL}$. 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 ℓth matrix from the first encoding with the ℓth 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 ℓth matrix of the output encoding. Again, intuitively, this works because:

1. For each $\ell \in [L]$, the ℓth 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 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

$$\mathbf{s} = \begin{bmatrix} 1 & 1 & 0 & F_1 & \dots & F_N \end{bmatrix} \mathbf{R}_0 \quad , \quad \mathbf{t} = \mathbf{R}_{NL}^{-1} \begin{bmatrix} 1 \\ 0 \\ 1 \\ G_1 \\ \vdots \\ G_N \end{bmatrix},$$

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 $(a, 1, 1, \dots, 1)$ 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.

On the Computational Assumption. In [MZ18], the authors based security on what amounted to a relatively straightforward version of the multilinear SXDH assumption, with the primary difference between their assumption and the textbook SXDH assumption being that their assumption is required to hold for arbitrary intervals. We define this formally later in the paper, but, for a reader familiar with the SXDH assumption over multilinear maps, the assumption is exactly what you might expect it to be.

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. To solve this, we introduce an assumption called "shifted"-SXDH, and we prove that this new "shifted"-SXDH assumption is at least as hard as the standard multilinear SXDH assumption in MZ18.

For some intiution, consider 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 first creating an encoding of k at level $[j + 1, j + 1]$ and then just invoking the MZ18 multiplication algorithm on these two elements. We note that this requires using a larger-degree multilinear map, and comment on the necessity of this later in the paper. Thus, 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.

Constructing Partial Slotted Weak PRFs. With the intuition above about how the MZ18 map works and the simple observation about how to multiply by a scalar (enabling us to have a PRF with key k), it is relatively straightforward to show an RKHwPRF construction. In a nutshell, we will use a series of "hash functions" $H_{i,j}$ that map arbitrary input to elements encoded in the interval $[i, j]$. These functions are defined in a somewhat contrived manner because we need the key homomorphism (i.e., the homomorphism with respect to the addition and multiplication operations between the key-space and output-space for the same input) to be consistent across different levels.

To compute the (weak) PRF, we "pair" these encoded elements corresponding to the inputs with an element k (representing the PRF key) encoded at an adjacent level. Note that due to certain structural restrictions inherent to the MZ18 MMap construction, this requires us to use two levels of the multilinear map in order to encode a single element. The homomorphic properties and the zero-test of this partial slotted RKHwPRF construction follow almost immediately from the definition of the MZ18 functions, so we defer them to the formal definition of our construction.

3 Preliminaries

This section introduces notations and presents preliminary background material.

3.1 Notation

For any positive integer n, we use $[n]$ to denote the set $\{1, \dots, n\}$. For two positive integers m and n we denote the set $\{m, m + 1, \dots, n\}$ 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. We use "$\overset{c}{\approx}$" to denote two distributions that are computationally indistinguishable and "$\overset{s}{\approx}$" to denote two distributions that are statistically indistinguishable.

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

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

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 Cryptographic Primitives

We present preliminary background material on various cryptographic primitives.

3.2.1 (Symmetric) Cryptographic Primitives

We begin by recalling the definitions of weak pseudorandom functions (wPRFs) and (pseudorandom) synthesizers.

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 assume throughout the sets K, X, Y are parameterized by some security parameter λ, but omit mentioning this explicitly for succinctness). 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 any security parameter λ and any probabilistic polynomial time (PPT) adversary $\mathcal{A}$, 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 (again, we assume throughout the sets X, Y, Z are parameterized by some security parameter λ, but omit mentioning this explicitly for succinctness). Assume that $\mathbf{x} \leftarrow X^{\ell}$ and $\mathbf{y} \leftarrow Y^{m}$ are two uniformly chosen vectors, and let $\mathbf{Z} \leftarrow Z^{\ell\times m}$ be a uniformly chosen matrix for integer parameters $\ell, m \geq 2$. We say that S is a pseudorandom synthesizer if for any security parameter λ and any PPT adversary $\mathcal{A}$, we have

$$|\Pr[\mathcal{A}[S(\mathbf{x}, \mathbf{y})] = 1] - \Pr[\mathcal{A}[\mathbf{Z}] = 1]| \leq \mathrm{negl}(\lambda),$$

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

3.2.2 Homomorphic Primitives

We consider weak PRFs and (pseudorandom) synthesizers endowed with ring homomorphism [AMPR19, AMP19, AMP23]. 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 on bounded group-homomorphic cryptoprimitives, and [AMP23] for a discussion on how such a notion may be extended to the setting of ring-homomorphic cryptoprimitives. In the definitions below, all sets/groups/rings are implicitly assumed to be parameterized by the security parameter λ, unless explicitly specified otherwise.

Definition 3.1. (Ring Key-Homomorphic Weak PRF [AMP23].) A weak PRF family $F : K \times X \to Y$ is a Ring Key-Homomorphic 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

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

Definition 3.2. (Ring Input-Homomorphic Weak PRF [AMPR19, AMP19].) A weak PRF family $F : K \times X \to Y$ is a Ring Input-Homomorphic 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

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

Definition 3.3. (Ring-Embedded Homomorphic Synthesizer [AMP23].) 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

$$S(x, g_1 \oplus g_2) = S(x, g_1) \boxplus S(x, g_2)$$

Claim 3.4 (RKHwPRF/RIHwPRF implies RHS). *A ring-embedded homomorphic synthesizer (RHS) is implied by an RKHwPRF or an RIHwPRF (for which the input space does not depend on the choice of the key).*

The proof is immediate and not detailed separately.

Remark 3.5 (Abelian vs Non-Abelian Rings.). Throughout the paper, unless specified otherwise, we consider generic finite rings with multiplicative identity where the *additive group is abelian* and *the multiplicative group is non-abelian*. Note that, for any finite ring R with the multiplicative identity 1_R, the additive group of R is necessarily abelian (this follows from the basic axioms of a ring). However, none of our constructions and proofs require assuming that the multiplicative group of R is abelian, since we primarily rely on multiplications of matrices consisting of elements from R, which is inherently non-abelian.

3.2.3 Indistinguishability Obfuscation (iO)

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

Definition 3.6 (Indistinguishability Obfuscator [BGI+01]). A PPT algorithm Obf is an indistinguishability obfuscator for a circuit family $\mathcal{C}_\lambda$ with input space $\{0,1\}^{\ell(\lambda)}$ (where λ is the security parameter) 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:

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

3.2.4 Input-Activated Indistinguishability Obfuscation (iaiO)

We now present a concise definition of an abstraction called input-activated indistinguishability obfuscation (iaiO) adopted from [GLSW15]. We refer the reader to [GLSW15] for more details.

Let $\mathcal{P}_\lambda$ be a family of oblivious branching programs, and let $P = \{P_1, \ldots, P_T\}$ be a set of T programs, where for each $t \in [T]$, the program $P_t \in \mathcal{P}_\lambda$ is an oblivious branching program of depth L of the form

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

where for each $t \in [T]$, each $\ell \in [L]$, and each $b \in \{0,1\}$, $M_{t,\ell,b} \in \{0,1\}^{5\times 5}$ is a permutation matrix. Each program P_t for $t \in [T]$ takes as input vectors from $\{0,1\}^N$. We additionally define the following variables/notations:

- We let $\phi : [L] \to [N]$ denote a mapping such that for each $\ell \in [L]$, the index $i = \phi(\ell)$ indicates which variable x_i in an input $\mathbf{x} = (x_1, \ldots, x_N) \in \{0,1\}^N$ controls the ℓth level branch. This mapping is assumed to be publicly known throughout.
- Let $P_t = \{M_{t,\ell,0}, M_{t,\ell,1}\}_{\ell \in [L]}$ be an oblivious branching program, where for each $\ell \in [L]$ and each $b \in \{0,1\}$, $M_{\ell,b} \in \{0,1\}^{5\times 5}$ is a permutation matrix. Also, let $\mathbf{x} = (x_1, \ldots, x_N) \in \{0,1\}^N$ denote an input. Let

  $$Q = [Q_{i,j}]_{i,j\in[5]} = \prod_{\ell=1}^{L} M_{\ell, x_{\phi(\ell)}}.$$

  We say that $P_t(\mathbf{x}, \phi) = 1$ if and only if Q is the identity permutation matrix (i.e., equivalently, $Q_{i,j} = 0$ for every non-diagonal position $(i,j) \in [5] \times [5]$ such that $i \neq j$). Otherwise, $P_t(\mathbf{x}, \phi) = 1$. In the rest of the paper, for simplicity of notation, we use $P_t(\mathbf{x})$ to denote $P_t(\mathbf{x}, \phi)$, where the mapping ϕ is implicitly used for program evaluation.
- We let $\mathbf{G} \in \{0,1\}^{N\times T\times 2}$ denote the *input activation matrix*. Given an input $\mathbf{x} = (x_1, \ldots, x_N) \in \{0,1\}^N$, we say that the program P_t for $t \in [T]$ is *activated* by the input $\mathbf{x}$ if $\mathbf{G}_{i,t,x_i} = 1$ for *all* $i \in [n]$. We denote by $(i,j,b) \in [N] \times [T] \times \{0,1\}$ a (row, column, slot) triplet for the matrix $\mathbf{G}$.
- For a column $t \in [T]$ of $\mathbf{G}$, we define a Boolean function $f_t : \{0,1\}^N \to \{0,1\}$ as follows: for an input $\mathbf{x} = (x_1, \ldots, x_N) \in \{0,1\}^N$, $f_t(\mathbf{x}) = 1$ if $\mathbf{G}_{i,t,x_i} = 1$ for each $i \in [N]$; otherwise, $f_t(\mathbf{x}) = 0$. We say that a program P_t is active on an input $\mathbf{x} \in \{0,1\}^N$ if and only if $f_t(\mathbf{x}) = 1$.

Definition 3.7 (Input-Activated iO [GLSW15]). An input-activated iO scheme consists of a pair of algorithms $(\mathsf{Create}, \mathsf{Eval})$ where:

- $\mathsf{Create}(1^\lambda, P = \{P_1, \ldots, P_T\}, \mathbf{G}, \phi)$: The creation algorithm takes as input the security parameter λ, a set of oblivious branching programs $P = \{P_1, \ldots, P_T\}$, the activation matrix $\mathbf{G}$, and the map ϕ, and output an input-activated obfuscation P.
- $\mathsf{Eval}(\mathsf{P}, \mathbf{x}, \phi)$: The evaluation algorithm takes an input-activated obfuscation P, an input $\mathbf{x} \in \{0,1\}^n$, and the map $\phi : [L] \to [N]$, and outputs a bit $b \in \{0,1\}$.

Any iaiO scheme is expected to satisfy the notions of correctness and security listed below.

Correctness. For any set of L-depth oblivious branching programs $P = \{P_1, \ldots, P_T\}$, any input activation matrix $\mathbf{G} \in \{0,1\}^{N\times T\times 2}$, and any input $\mathbf{x} = (x_1, \ldots, x_N) \in \{0,1\}^N$, let the set $S_\mathbf{x}$ be defined as:

$$S_\mathbf{x} = \{t \in [T] : \mathbf{G}_{i,t,x_i} = 1 \quad \forall i \in [N]\}.$$

An iaiO scheme $(\mathsf{Create}, \mathsf{Eval})$ is said to be correct if for any security parameter λ, any set of L-depth oblivious branching programs $P = \{P_1, \ldots, P_T\}$, any input activation matrix $\mathbf{G} \in \{0,1\}^{N\times T\times 2}$, and any input $\mathbf{x} = (x_1, \ldots, x_N) \in \{0,1\}^N$, whenever $S_\mathbf{x} \neq \emptyset$ and $P_t = P_{t'}$ for all $t, t' \in S_\mathbf{x}$, then, letting $\mathsf{P} \leftarrow \mathsf{Create}(1^\lambda, P = \{P_1, \ldots, P_T\}, \mathbf{G}, \phi)$, the following holds for each $t \in S_\mathbf{x}$:

$$\mathsf{Eval}(\mathsf{P}, \mathbf{x}, \phi) = P_t(\mathbf{x}).$$

Experiment $\mathsf{Expt}^{\text{Inter-column Security}}_{\lambda,\mathcal{A}}$:

- The adversary $\mathcal{A}$ provides to the challenger a set of L-depth oblivious branching programs $P = \{P_1, \ldots, P_T\}$, an input activation matrix $\mathbf{G} \in \{0,1\}^{N \times T \times 2}$, a map $\phi : [L] \to [N]$, two indices $t_0, t_1 \in [T]$, an index $i^* \in [N]$, and $\beta \in \{0,1\}$ subject to the following restrictions:
  1. The input activation matrix entry $\mathbf{G}_{i^*,t_0,\beta} = 1$.
  2. The program descriptions P_{t_0} and P_{t_1} are identical.
  3. For each row-slot pair $(i,\gamma) \in [N] \times \{0,1\}$ except for the row-slot pair $(i^*, 1-\beta)$, if $\mathbf{G}_{i,t_1,\gamma} = 1$ then $\mathbf{G}_{i,t_0,\gamma} = 1$ (this essentially means that the column t_0 "dominates" column t_1).
- The game between the challenger and $\mathcal{A}$ now proceeds as follows:
  1. The challenger samples a bit $b \leftarrow \{0,1\}$.
  2. If $b = 0$, the challenger runs $\mathsf{P} \leftarrow \mathsf{Create}(1^\lambda, P, \mathbf{G}, \phi)$.
  3. If $b = 1$, the challenger first creates $\mathbf{G}'$ by copying $\mathbf{G}$ and flipping the entry $G_{i^*,t_1,\beta}$, and then runs $\mathsf{P} \leftarrow \mathsf{Create}(1^\lambda, P, \mathbf{G}', \phi)$.
  4. The challenger then gives P to $\mathcal{A}$.
  5. Eventually, $\mathcal{A}$ outputs $b' \in \{0,1\}$.

Figure 2: Inter-Column Security Game for any iaiO Scheme.

Inter-column Security. We define inter-column security of an iaiO scheme $(\mathsf{Create}, \mathsf{Eval})$ using the security game $\mathsf{Expt}^{\text{Inter-column Security}}_{\lambda,\mathcal{A}}$ parameterized by the security parameter λ as defined in Figure 2. We say that the iaiO scheme $(\mathsf{Create}, \mathsf{Eval})$ satisfies inter-column security if for any security parameter λ and any PPT adversary $\mathcal{A}$ that outputs b' in the inter-column security game, we have

$$\left| \Pr[b' = b] - \frac{1}{2} \right| \leq \mathrm{negl}(\lambda).$$

Intra-column Security. Next, we define intra-column security of an iaiO scheme $(\mathsf{Create}, \mathsf{Eval})$ using the security game $\mathsf{Expt}^{\text{Intra-column Security}}_{\lambda,\mathcal{A}}$ parameterized by the security parameter λ as defined in Figure 3. We say that the iaiO scheme $(\mathsf{Create}, \mathsf{Eval})$ satisfies intra-column security if for any security parameter λ and any PPT adversary $\mathcal{A}$ that outputs b' in the intra-column security game, we have

$$\left| \Pr[b' = b] - \frac{1}{2} \right| \leq \mathrm{negl}(\lambda).$$

Completely Inactive Program Security. Next, we define completely inactive program security of an iaiO scheme $(\mathsf{Create}, \mathsf{Eval})$ using the security game $\mathsf{Expt}^{\text{Completely Inactive Program Security}}_{\lambda,\mathcal{A}}$ parameterized by the security parameter λ as defined in Figure 4. We say that the iaiO scheme $(\mathsf{Create}, \mathsf{Eval})$ satisfies completely inactive program security if for any security parameter λ and any PPT adversary $\mathcal{A}$ that outputs b' in the complete inactive program security game, we have

$$\left| \Pr[b' = b] - \frac{1}{2} \right| \leq \mathrm{negl}(\lambda).$$

Experiment $\mathsf{Expt}^{\text{Intra-column Security}}_{\lambda,\mathcal{A}}$:

- The PPT adversary $\mathcal{A}$ provides to the challenger a set of L-depth oblivious branching programs $P = \{P_1, \ldots, P_T\}$, an input activation matrix $\mathbf{G} \in \{0,1\}^{N \times T \times 2}$, a map $\phi : [L] \to [N]$, and three indices $t^*, t_0, t_1 \in [T]$ subject to the following restrictions:
  1. The input activation matrix entries $\mathbf{G}_{i^*,t^*,0} = \mathbf{G}_{i^*,t^*,1} = 0$.
  2. The input activation matrix entries $\mathbf{G}_{i^*,t_0,0} = \mathbf{G}_{i^*,t_0,1} = 0$.
  3. The input activation matrix entries $\mathbf{G}_{i^*,t_1,0} = \mathbf{G}_{i^*,t_1,1} = 0$.
- The game between the challenger and $\mathcal{A}$ now proceeds as follows:
  1. The challenger samples a bit $b \leftarrow \{0,1\}$.
  2. If $b = 0$, the challenger runs $\mathsf{P} \leftarrow \mathsf{Create}(1^\lambda, P, \mathbf{G}, \phi)$.
  3. If $b = 1$, the challenger first creates $\mathbf{G}'$ by copying $\mathbf{G}$ except it sets $\mathbf{G}'_{i,t^*,b} = \mathbf{G}_{i,t_b,b}$ for each $i \in [N]$ and $b \in \{0,1\}$, and then runs $\mathsf{P} \leftarrow \mathsf{Create}(1^\lambda, P, \mathbf{G}', \phi)$.
  4. The challenger then gives P to $\mathcal{A}$.
  5. Eventually, $\mathcal{A}$ outputs $b' \in \{0,1\}$.

Figure 3: Intra-Column Security Game for any iaiO Scheme.

Experiment $\mathsf{Expt}^{\text{Completely Inactive Program Security}}_{\lambda,\mathcal{A}}$:

- The PPT adversary $\mathcal{A}$ provides to the challenger a set of L-depth oblivious branching programs $P = \{P_1, \ldots, P_T\}$, an input activation matrix $\mathbf{G} \in \{0,1\}^{N \times T \times 2}$, a map $\phi : [L] \to [N]$, an alternate oblivious branching program P^*, and an index $t^* \in [T]$ subject to the following restriction: the input activation matrix entries $\mathbf{G}_{i,t^*,b} = 0$ for all $i \in [N]$ and $b \in \{0,1\}$.
- The game between the challenger and $\mathcal{A}$ now proceeds as follows:
  1. The challenger samples a bit $b \leftarrow \{0,1\}$.
  2. If $b = 0$, the challenger runs $\mathsf{P} \leftarrow \mathsf{Create}(1^\lambda, P, \mathbf{G}, \phi)$.
  3. If $b = 1$, the challenger first creates a set P' by copying P except it replaces the original program P_{t^*} by the alternate program P^*, and then runs $\mathsf{P} \leftarrow \mathsf{Create}(1^\lambda, P', \mathbf{G}, \phi)$.
  4. The challenger then gives P to $\mathcal{A}$.
  5. Eventually, $\mathcal{A}$ outputs $b' \in \{0,1\}$.

Figure 4: Completely Inactive Program Security Game for any iaiO Scheme.

Single-input Program Security. Finally, we define single-input program security of an iaiO scheme $(\mathsf{Create}, \mathsf{Eval})$ using the security game $\mathsf{Expt}^{\text{Single-Input Program Security}}_{\lambda,\mathcal{A}}$ parameterized by the security parameter λ as defined in Figure 5. We say that the iaiO scheme $(\mathsf{Create}, \mathsf{Eval})$ satisfies single-input program security if for any security parameter λ and any PPT adversary $\mathcal{A}$ that outputs b' in the single-input program security game, we have

$$\left| \Pr[b' = b] - \frac{1}{2} \right| \leq \mathrm{negl}(\lambda).$$

Experiment $\mathsf{Expt}^{\text{Single-Input Program Security}}_{\lambda,\mathcal{A}}$:

- The PPT adversary $\mathcal{A}$ provides to the challenger a set of L-depth oblivious branching programs $P = \{P_1, \ldots, P_T\}$, an input activation matrix $\mathbf{G} \in \{0,1\}^{N \times T \times 2}$, a map $\phi : [L] \to [N]$, an alternate oblivious branching program P^*, and an index $t^* \in [T]$ subject to the following restriction: there exists an input $\mathbf{x}^* \in \{0,1\}^N$ such that:
  1. The t^*-th column of $\mathbf{G}$ corresponds to a point function f_{t^*} where f_{t^*} evaluates to 1 on the single input x^*.
  2. $P_{t^*}(x^*) = P^*(x^*)$.
- The game between the challenger and $\mathcal{A}$ now proceeds as follows:
  1. The challenger samples a bit $b \leftarrow \{0,1\}$.
  2. If $b = 0$, the challenger runs $\mathsf{P} \leftarrow \mathsf{Create}(1^\lambda, P, \mathbf{G}, \phi)$.
  3. If $b = 1$, the challenger first creates a set P' by copying P except it replaces the original program P_{t^*} by the alternate program P^*, and then runs $\mathsf{P} \leftarrow \mathsf{Create}(1^\lambda, P', \mathbf{G}, \phi)$.
  4. The challenger then gives P to $\mathcal{A}$.
  5. Eventually, $\mathcal{A}$ outputs $b' \in \{0,1\}$.

Figure 5: Single-Input Program Security Game for any iaiO Scheme.

3.3 Leftover Hash Lemma

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

Lemma 3.8 (Leftover Hash Lemma (LHL) [IZ89]). *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 (note that we *do not* require the multiplicative group of the ring to be abelian).

Lemma 3.9 (LHL over Rings). *Let R be a finite ring with additive/multiplicative identity $0_R/1_R$ such that $|R| = \omega(\lambda^c)$, 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*

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

where $u \leftarrow R$ is a uniformly chosen ring element and $\mathbf{s} \leftarrow \{0_R, 1_R\}^m$.

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

Lemma 3.10 (LHL over Ideals [Mic02]). *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 $(r_1, \ldots, r_m)$.*

3.4 Some Results on RHS

In this subsection, we recall from [AMP23] certain hardness assumptions that are implied by any generic RHS. Looking ahead, we use these hardness assumptions crucially for our iO construction. We begin by stating the following theorem, which is essentially an adaptation of Theorem 1 from [AMP19] to the setting of ring-embedded homomorphic synthesizers. We also include the proof for the sake of completeness.

Theorem 3.11 (Imported from [AMP23]). *Let $S : X \times G \to R$ be a ring-embedded homomorphic synthesizer where $|X| > \omega(\lambda^c)$ and $|G| > \omega(\lambda^c)$ for any arbitrary constant c, and let $m = \mathrm{poly}(\lambda)$ be an (arbitrary) positive integer. Let $d = \mathrm{poly}(\lambda)$ be such that $d > 3\log|G|$. Let $\mathbf{R} \leftarrow R^{m\times d}$ be a 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*

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

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 $(\mathbf{M}, \mathbf{Ms}) \overset{c}{\approx} (\mathbf{R}, \mathbf{u})$ 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 3.9) we can write

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

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

We state an additional theorem below. The proof is essentially identical to the proof of Theorem 3.11, and is hence not detailed separately.

Theorem 3.12. *Let $S : X \times G \to R$ be a ring-embedded homomorphic synthesizer where $|X| > \omega(\lambda^c)$ and $|G| > \omega(\lambda^c)$ for any arbitrary constant c, and let $m = \mathrm{poly}(\lambda)$ be an (arbitrary) positive integer. Let $d = \mathrm{poly}(\lambda)$ be such that $d > 3\log|G|$. Let $\mathbf{R} \leftarrow R^{m\times d}$ be a 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*

$$(\mathbf{R}, \mathbf{sR}) \overset{c}{\approx} (\mathbf{R}, \mathbf{u}^t),$$

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

We recall an additional theorem from [AMP23], and again include the proof for the sake of completeness.

Theorem 3.13 (Imported from [AMP23]). *Let $S : X \times G \to R$ be a ring-embedded homomorphic synthesizer where $|X| > \omega(\lambda^c)$ and $|G| > \omega(\lambda^c)$ for any arbitrary constant c, and let $m = \mathrm{poly}(\lambda)$ 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 $F : M_m(R) \times M_m(R) \to M_m(R)$ is 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*

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

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 $Q = \mathrm{poly}(\lambda)$ be any arbitrary positive integer. It is enough to show that

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

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 3.11, 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

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

Finally, we state another additional theorem below. The proof is essentially identical to the proof of Theorem 3.13 (with the only difference being that we apply Theorem 3.12 instead of Theorem 3.11), and is hence not detailed separately.

Theorem 3.14. *Let $S : X \times G \to R$ be a ring-embedded homomorphic synthesizer where $|X| > \omega(\lambda^c)$ and $|G| > \omega(\lambda^c)$ for any arbitrary constant c, and let $m = \mathrm{poly}(\lambda)$ 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 $F : M_m(R) \times M_m(R) \to M_m(R)$ is the function defined by $F(\mathbf{K}, \mathbf{X}) = \mathbf{K} \boxtimes \mathbf{X}$, then F is a weak PRF (and hence a synthesizer). In addition, F satisfies (left) $M_m(R)$-module homomorphism over the key space, i.e., for any $\mathbf{K}, \mathbf{K}', \mathbf{X} \in M_m(R)$ we have*

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

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

The following corollary follows immediately from Theorems 3.13 and 3.14.

Corollary 3.15. *Let $S : X \times G \to R$ be a ring-embedded homomorphic synthesizer where $|X| > \omega(\lambda^c)$ and $|G| > \omega(\lambda^c)$ for any arbitrary constant c. Let $m = \mathrm{poly}(\lambda)$ be a positive integer such that $m > 3\log|G|$, and let $Q = \mathrm{poly}(\lambda)$ be any arbitrary positive integer. Then we have*

$$(\mathbf{A}, \mathbf{AK}) \overset{c}{\approx} (\mathbf{A}, \mathbf{U}), \quad (\mathbf{B}, \mathbf{K}'\mathbf{B}) \overset{c}{\approx} (\mathbf{B}, \mathbf{V})$$

where $\mathbf{K}, \mathbf{K}' \leftarrow R^{m \times m}$, $\mathbf{A}, \mathbf{U} \leftarrow R^{Qm \times m}$, and $\mathbf{B}, \mathbf{V} \leftarrow R^{m \times Qm}$.

4 Input-Activated Indistinguishability Obfuscator from RHS

In this section, we show how to construct an (input-activated) indistinguishability obfuscator for $\mathcal{NC}^1$ from any ring-embedded homomorphic synthesizer (RHS). We begin by fixing some notations used throughout this section.

Notations. Given an RHS $S : X \times G \to R$, we fix an appropriately large parameter m (the concrete choice of m will be discussed subsequently in Section 5). Let $\mathbf{I}$ and $\mathbf{0}$ denote the identity matrix of dimension $m \times m$ and the all-zero matrix of dimension $m \times m$ over the ring R, respectively.

4.1 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 ℓth level branch.

Kilian-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.

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 $N_{\ell,b} \in \{0,1\}^{5\times 5}$, 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:

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

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 $c_1 m \times c_2 m$ and the "right guard" $\mathbf{R}_\ell$ is of dimension $c_2 m \times c_1 m$ where c_1, c_2 are integers such that $c_1 \gg c_2$. 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

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

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

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

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:

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

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

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

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 $c_1 \gg 5$.

3. **Step-3:** Generate $2L$ "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 a matrix of dimension $5m \times 5m$. Suppose, $\mathbf{Q}$ is structured as follows:

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

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

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

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

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

The zero test will then choose a particular non-diagonal entry (i, j) that 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 sub-matrix $\mathbf{Q}_{i,j}$ 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.

Intuitively, the program-carrying matrices are ring encodings of the matrices of the associated branching programs, while the enforcer matrices enable enforcing that the input bits used in the evaluation of the branching programs are consistent. We now describe in details how each of the above 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:

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

For each level $\ell \in [L]$ and for each bit $b \in \{0,1\}$, we construct a "program-carrying matrix" $\mathbf{P}_{\ell,b}$ of dimension $(5m(2L+1)) \times (5m(2L+1))$, structured as:

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

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

$$\mathbf{C}_{(\ell,\ell'),(b,b')} = \begin{cases} \mathbf{0}_R & \text{if } (\ell', b') = (\ell, 1-b), \\ \text{uniform in } R^{5m \times 5m} & \text{otherwise.} \end{cases}.$$

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 matrix" $\mathbf{E}_{i,b}$ of dimension $(5m(2L+1)) \times (5m(2L+1))$, structured as:

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

where for each $i \in [N]$, and each $b \in \{0,1\}$, $\mathbf{T}_{i,b}$ is a uniform *block-diagonal* matrix in $R^{5m \times 5m}$, and also for each $i \in [N]$, each $\ell' \in [L]$, and each $b, b' \in \{0,1\}$, we have

$$\mathbf{C}_{(i,\ell'),(b,b')} = \begin{cases} \mathbf{0}_R & \text{if } (i, b') = (\phi(\ell'), b), \\ \text{uniform in } R^{5m \times 5m} & \text{otherwise.} \end{cases},$$

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 ℓth level branch.

Remark 4.1. While the consistency sub-matrices have (semantically) different roles in the program-carrying and enforcer matrices, we use the same notation to denote them for simplicity of exposition.

3. **Guarding Program-Carrying Matrices 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 matrices and the 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 $c_2 = 5(2L+1)$ and $c_1 \gg 5(2L+1)$.

   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 $\mathbf{L}_1 \in R^{5m \times 5m(2L+1)}$ and $\mathbf{R}_{L+N} \in R^{5m(2L+1) \times 5m}$ are structured as follows:

$$\mathbf{L}_1 = \begin{bmatrix} \mathbf{I} & \mathbf{I} & \dots & \mathbf{I} \end{bmatrix}, \quad \mathbf{R}_{L+N} = \begin{bmatrix} \mathbf{I} \\ \mathbf{I} \\ \vdots \\ \mathbf{I} \end{bmatrix},$$

where we overload the notation $\mathbf{I}$ to denote the identity matrix over $R^{5m \times 5m}$.

Finally, we create a sequence of $2L$ "guarded" program matrix encodings and $2N$ enforcer matrix encodings of the form

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

where for each level $\ell \in [L]$ and each bit $b \in \{0,1\}$, we have

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

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 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:

$$\mathbf{Q} = \left(\prod_{\ell=1}^{L} \widetilde{\mathbf{P}}_{\ell, x_{\phi(\ell)}} \right) \left(\prod_{i=1}^{N} \widetilde{\mathbf{E}}_{i, x_i} \right).$$

Note that because of the structuring of the end-guard matrices, $\mathbf{Q} = [\mathbf{Q}_{i,j}]_{i,j \in [5]}$ is a $5m \times 5m$ matrix of elements in the ring R, where each $\mathbf{Q}_{i,j} \in R^{m \times m}$. 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

$$Q = \prod_{\ell=1}^{L} M_{\ell, x_{\phi(\ell)}} := \begin{bmatrix} Q_{1,1} & \cdots & Q_{1,5} \\ \vdots & \ddots & \vdots \\ Q_{5,1} & \cdots & Q_{5,5} \end{bmatrix} \in \{0,1\}^{5 \times 5}.$$

As in the simple construction, the zero test will choose a single non-diagonal matrix position (i, j) so that the plaintext entry $Q_{i,j}$ is non-zero in the subset-product matrix Q whenever the branching program outputs 0 (i.e. whenever the matrix product Q is not the identity permutation matrix). At this point, the zero test simply checks if $\mathbf{Q}_{i,j} = \mathbf{0}$. 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 $(\ell, b) \in [L] \times \{0,1\}$, if $b \neq x_\phi(\ell)$, then the consistency sub-matrices in the corresponding diagonal slot do not contribute to the final product since there is a zero sub-matrix in this diagonal slot in the program carrying matrix $\mathbf{P}_{\ell, x_{\phi(\ell)}}$.
2. For each $(\ell, b) \in [L] \times \{0,1\}$, if $b = x_\phi(\ell)$, then the consistency sub-matrices in the corresponding diagonal slot do not contribute to the final product since there is a zero sub-matrix in this diagonal slot in the enforcer matrix $\mathbf{E}_{\phi(\ell), x_{\phi(\ell)}}$.

Thus the only potential contributions to the value of $\mathbf{Q}_{i,j}$ come from the diagonal slots corresponding to the ring-embedded permutation matrices. It is now easy to see that $\mathbf{Q}_{i,j} \neq \mathbf{0}$ 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 $\mathbf{x}$.

4.2 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-embedded 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

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

Next, as in the core iO construction, for each $t \in [T]$, we uniformly sample Kilian randomizer matrices $Z_{t,1}, \ldots, Z_{t,L-1} \in \{0,1\}^{5\times 5}$, and define a new set of matrices $\{N_{t,\ell,b}\}_{t\in[T],\ell\in[L],b\in\{0,1\}}$, where

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

Note that for each $t \in [T]$, both $Z_{t,0}$ and $Z_{t,L}$ are set to be the identity permutation.

Next, as in the core iO construction, 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 $\mathbf{N}_{t,\ell,b}$ of dimension $5m \times 5m$. In particular, for each $(w,v) \in [5] \times [5]$, we have $\mathbf{N}_{t,\ell,b,w,v}$ is either the all-zero matrix in $R^{m\times m}$ (if the corresponding plaintext entry $N_{t,\ell,b,w,v}$ is 0) or a uniformly randomly sampled matrix in $R^{m\times m}$ (if the corresponding plaintext entry $N_{t,\ell,b,w,v}$ is 1).

Now, for each level $\ell \in [L]$ and each bit $b \in \{0,1\}$, we construct an "aggregate" block-diagonal matrix $\mathbf{A}_{\ell,b}$ of dimension $5Tm \times 5Tm$ structured as:

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

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 "program-carrying matrix" $\mathbf{P}'_{\ell,b}$, with the only difference being that it is now of dimension $(5m(T+2L)) \times (5m(T+2L))$, and is structured as:

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

where the consistency sub-matrices are generated exactly as in the core iO construction.

Note that the only change from the core iO construction is that we now place the "aggregate matrix" $\mathbf{A}_{\ell,b}$ 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". 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 that $\mathbf{T}_{i,b}$ is a block-diagonal matrix sampled uniformly from $R^{5m\times 5m}$. In the parallel version of the construction, we instead structure $\mathbf{T}_{i,b}$ as a block diagonal matrix in $R^{5Tm\times 5Tm}$, 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 a block-diagonal matrix sampled uniformly from $R^{5m\times 5m}$. Now, as in the core iO constriction, we construct the enforcer matrix $\mathbf{E}'_{i,b}$ of dimension $(5m(T + 2L)) \times (5m(T + 2L))$ as:

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

where the consistency sub-matrices are generated exactly as in the core iO construction.

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 $c_2 = 5(T + 2L)$ and $c_1 \gg 5(T + 2N)$. In particular, the "end-guard" matrices $\mathbf{L}_1 \in R^{5m\times 5m(T+2L)}$ and $\mathbf{R}_{L+N} \in R^{5m(T+2L)\times 5m}$ are again structured as follows:

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

Finally, as in the core iO construction, we create a sequence of $2L$ "guarded" program matrix encodings and $2N$ enforcer matrix encodings of the form

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

where for each level $\ell \in [L]$, each bit $b \in \{0, 1\}$, we have

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

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, Zero-testing, Correctness. Evaluation and zero-testing for the parallel iO construction work exactly as in the core iO construction. To see that correctness is ensured, 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 sub-matrices in the corresponding diagonal slot do not contribute to the final product since there is a zero sub-matrix in this diagonal slot in the program carrying matrix $\mathbf{P}_{\ell, x_{\phi(\ell)}}$.
2. For each (ℓ, b), if $b = x_\phi(\ell)$, then the consistency sub-matrices in the corresponding diagonal slot do not contribute to the final product since there is a zero sub-matrix 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 $\mathbf{Q}_{i,j}$ come from the diagonal slots corresponding to the ring-embedded permutation matrices. It is now easy to see that $\mathbf{Q}_{i,j} \neq \mathbf{0}$ 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 $\mathbf{x}$.

4.3 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 [GLSW15].

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 $5 \cdot (T + 2(L + N)) \cdot m \times 5 \cdot (T + 2(L + N)) \cdot m$, structured as:

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

In particular, we structured $\mathbf{T}_{i,b}$ as a block diagonal matrix in $R^{5Tm \times 5Tm}$:

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

where for each program index $t \in [T]$, the block diagonal matrix $\mathbf{T}_{t,i,b}$ was sampled uniformly from $R^{5m \times 5m}$. In the full-fledged iaiO construction, we generate these enforcer matrices in exactly the same way, except for the manner in which each $\mathbf{T}_{t,i,b}$ block matrix is generated. We generate each $\mathbf{T}_{t,i,b}$ block diagonal 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:

$$\mathbf{T}_{t,i,b} = \begin{cases} \mathbf{0}_R & \text{if } \mathbf{G}[i, t, b] = 0, \\ \text{uniform block-diagonal in } R^{5m \times 5m} & \text{otherwise.} \end{cases}.$$

Evaluation, Zero-testing, Correctness. Evaluation and zero-testing for the iaiO construction are exactly the same as in the parallel iO construction. To see that correctness is maintained, we first observe that the consistency-check sub-matrices do not contribute to the result. This is the same as in the previous construction with the with the all-ones input activation matrix. Additionally, observe the following:

$\mathsf{GenAggMatrix}_R(1^m, 1^T, \{N_t \in \{0,1\}^{5\times 5}\}_{t\in[T]})$:

1. Create a set of ring-embedded permutation matrices $\{\mathbf{N}_t\}_{t\in[T]}$, where each $\mathbf{N}_t$ of dimension $5m \times 5m$ is structured as follows:
$$\mathbf{N}_t = \begin{bmatrix} \mathbf{N}_{t,1,1} & \dots & \mathbf{N}_{t,1,5} \\ \vdots & \ddots & \vdots \\ \mathbf{N}_{t,5,1} & \dots & \mathbf{N}_{t,5,5} \end{bmatrix}.$$
such that, for each $(w, v) \in [5] \times [5]$, we have $\mathbf{N}_{t,w,v}$ is either the all-zero matrix in $R^{m\times m}$ (if the corresponding plaintext entry $N_{t,w,v}$ is 0) or a uniformly randomly sampled matrix in $R^{m\times m}$ (if the corresponding plaintext entry $N_{t,w,v}$ is 1).
2. Construct an "aggregate" block-diagonal matrix $\mathbf{A} \in R^{5Tm\times 5Tm}$ structured as:
$$\mathbf{A} = \begin{bmatrix} \mathbf{N}_1 & & \\ & \ddots & \\ & & \mathbf{N}_T \end{bmatrix}.$$
3. Output $\mathbf{A}$.

Figure 6: Algorithm to Generate Aggregate Matrices

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 $\mathbf{T}$ sub-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 $\mathbf{T}$ sub-matrices will prevent the program P_t from contributing to the final output.

It is now easy to see that $\mathbf{Q}_{i,j} \neq \mathbf{0}$ 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 $\mathbf{x}$.

4.4 Summary: The Final iaiO Construction

In this subsection, we put everything together and summarize our final iaiO construction, which is a tuple of algorithms $(\mathsf{Create}, \mathsf{Eval})$ described as follows:

- $\mathsf{Create}(1^\lambda, \mathbf{G}, (P_1, \dots, P_T), \phi)$: The Create algorithm takes as input the security parameter λ, the activation matrix $\mathbf{G}$, the set of oblivious branching programs $P_1, \dots, P_T$, and a map ϕ where:
  - $\mathbf{G}$ is the activation matrix of dimension $N \times T \times 2$, where each $i \in [N]$ represents the row corresponding to the ith input bit, each $t \in [T]$ represents the column corresponding to the program P_t, and $b \in \{0, 1\}$ is the bit corresponding to the activation on the particular input bit value.
  - For each $t \in [T]$, P_t is an oblivious branching program of depth L of the form
$$P_t = \{M_{t,\ell,0}, M_{t,\ell,1}\}_{\ell\in[L]}.$$
  - $\phi : [L] \to [N]$ is a mapping such that for each $\ell \in [L]$, $i = \phi(\ell)$ indicates which variable x_i controls the ℓth level branch.

  The Create algorithm samples an RHS $S : X \times G \to R$ according to the security parameter λ and fixes an appropriately large parameter m (the concrete choice of m will be discussed subsequently in Section 5). It then proceeds as follows:

$\mathsf{GenProgCarryingMatrix}_R(1^m, 1^T, 1^L, 1^N, \mathbf{A} \in R^{5Tm \times 5Tm}, (\ell, b) \in [L] \times \{0,1\})$:

1. Construct $\mathbf{P} \in R^{(5m(T+2L)) \times (5m(T+2L))}$, structured as:

$$\mathbf{P} = \begin{bmatrix} \mathbf{A} & & & & & & \\ & \mathbf{C}_{1,0} & & & & & \\ & & \mathbf{C}_{1,1} & & & & \\ & & & \ddots & & & \\ & & & & \mathbf{C}_{L,0} & & \\ & & & & & \mathbf{C}_{L,1} & \end{bmatrix},$$

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

$$\mathbf{C}_{\ell',b'} = \begin{cases} \mathbf{0}_R \in R^{5m \times 5m} & \text{if } (\ell', b') = (\ell, 1-b), \\ \text{uniform in } R^{5m \times 5m} & \text{otherwise.} \end{cases}.$$

2. Output $\mathbf{P}$.

Figure 7: Algorithm to Generate Program-Carrying Matrices

1. For $t \in [T]$, sample uniform Kilian randomizers $Z_{t,1}, \ldots, Z_{t,L-1} \in \{0,1\}^{5\times 5}$. For each $t \in [T]$, $\ell \in [L]$ and $b \in \{0,1\}$, set
$$N_{t,\ell,b} = Z_{t,\ell-1}^{-1} M_{t,\ell,b} Z_{t,\ell},$$
where, for each $t \in [T]$, both $Z_{t,0}$ and $Z_{t,L}$ are set to be the identity permutation.
2. For each level $\ell \in [L]$ and each bit $b \in \{0,1\}$, generate an aggregate block-diagonal matrix $\mathbf{A}_{\ell,b} \in R^{5Tm \times 5Tm}$ as
$$\mathbf{A}_{\ell,b} \leftarrow \mathsf{GenAggMatrix}_R(1^m, 1^T, \{N_{t,\ell,b}\}_{t\in[T]}),$$
where the algorithm $\mathsf{GenAggMatrix}_R$ (parameterized by the ring R) is as described in Figure 6.
3. For each level $\ell \in [L]$ and for each bit $b \in \{0,1\}$, generate a program-carrying matrix $\mathbf{P}_{\ell,b} \in R^{(5m(T+2L)) \times (5m(T+2L))}$ as
$$\mathbf{P}_{\ell,b} \leftarrow \mathsf{GenProgCarryingMatrix}_R(1^m, 1^T, 1^L, 1^N, \mathbf{A}_{\ell,b}, (\ell, b)),$$
where the algorithm $\mathsf{GenProgCarryingMatrix}_R$ (parameterized by the ring R) is as described in Figure 7.
4. For each variable index $i \in [N]$ and for each bit $b \in \{0,1\}$, generate an enforcer matrix $\mathbf{E}_{i,b} \in R^{(5m(T+2L)) \times (5m(T+2L))}$ as
$$\mathbf{E}_{i,b} \leftarrow \mathsf{GenEnforcerMatrix}_R(1^m, 1^T, 1^L, 1^N, \mathbf{G}, (i, b), \phi)$$
where the algorithm $\mathsf{GenEnforcerMatrix}_R$ (parameterized by the ring R) is as defined in Figure 8, and where $\phi : [L] \to [N]$ is a mapping such that for each $\ell \in [L]$, $i = \phi(\ell)$ indicates which variable x_i controls the ℓth level branch.
5. Generate a sequence of guard matrices
$$\{(\mathbf{L}_\ell, \mathbf{R}_\ell)\}_{\ell \in [L+N]} \leftarrow \mathsf{GenGuardMatrices}_R(1^m, 1^T, 1^L, 1^N),$$
where the algorithm $\mathsf{GenGuardMatrices}_R$ (parameterized by the ring R) is as described in Figure 9.
6. Create a sequence of $2L$ "guarded" program matrix encodings and $2N$ enforcer matrix encodings of the form
$$\{\widetilde{\mathbf{P}}_{\ell,b}\}_{\ell\in[L],b\in\{0,1\}}, \quad \{\widetilde{\mathbf{E}}_{i,b}\}_{i\in[N],b\in\{0,1\}},$$

$\mathsf{GenEnforcerMatrix}_R(1^m, 1^T, 1^L, 1^N, \mathbf{G} \in \{0,1\}^{N \times T \times 2}, (i,b) \in [N] \times \{0,1\}, \phi : [L] \to [N])$:

1. Construct a block diagonal matrix in $\mathbf{T} \in R^{5Tm \times 5Tm}$ as

$$\mathbf{T} = \begin{bmatrix} \mathbf{T}_1 & & \\ & \cdots & \\ & & \mathbf{T}_T \end{bmatrix},$$

where for each program index $t \in [T]$, the matrix $\mathbf{T}_t$ is generated as follows:

$$\mathbf{T}_t = \begin{cases} \mathbf{0}_R \in R^{5m \times 5m} & \text{if } \mathbf{G}[i,t,b] = 0, \\ \text{uniform block-diagonal in } R^{5m \times 5m} & \text{otherwise.} \end{cases}.$$

2. Construct $\mathbf{E} \in R^{(5m(T+2L)) \times (5m(T+2L))}$, structured as:

$$\mathbf{E} = \begin{bmatrix} \mathbf{T} & & & & & & \\ & \mathbf{C}_{1,0} & & & & & \\ & & \mathbf{C}_{1,1} & & & & \\ & & & \ddots & & & \\ & & & & \mathbf{C}_{L,0} & & \\ & & & & & \mathbf{C}_{L,1} & \end{bmatrix},$$

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

$$\mathbf{C}_{\ell',b'} = \begin{cases} \mathbf{0}_R \in R^{5m \times 5m} & \text{if } (i,b') = (\phi(\ell), b), \\ \text{uniform in } R^{5m \times 5m} & \text{otherwise.} \end{cases},$$

3. Output $\mathbf{E}$.

Figure 8: Algorithm to Generate Enforcer Matrices

where for each level $\ell \in [L]$, each bit $b \in \{0,1\}$, we have

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

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

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

7. Output $\mathsf{P} = \left(\{\widetilde{\mathbf{P}}_{\ell,b}\}_{\ell \in [L], b \in \{0,1\}}, \{\widetilde{\mathbf{E}}_{i,b}\}_{i \in [N], b \in \{0,1\}} \right)$.

- $\mathsf{Eval}(\mathsf{P}, \mathbf{x} \in \{0,1\}^n, \phi)$: On input the obfuscated program

$$\mathsf{P} = \left(\{\widetilde{\mathbf{P}}_{\ell,b}\}_{\ell \in [L], b \in \{0,1\}}, \{\widetilde{\mathbf{E}}_{i,b}\}_{i \in [N], b \in \{0,1\}} \right),$$

an input vector $\mathbf{x} = (x_1, \ldots, x_N) \in \{0,1\}^n$ and the map $\phi : [L] \to [N]$, the Eval algorithm proceeds as follows:

1. Compute $\mathbf{Q} = \left(\prod_{\ell=1}^{L} \widetilde{\mathbf{P}}_{\ell, x_{\phi(\ell)}} \right) \left(\prod_{i=1}^{N} \widetilde{\mathbf{E}}_{L+i, x_i} \right)$.
2. Parse $\mathbf{Q} \in R^{5m \times 5m}$ as $\mathbf{Q} = [\mathbf{Q}_{i,j}]_{i,j \in [5]}$, where each $\mathbf{Q}_{i,j} \in R^{m \times m}$.

$\mathsf{GenGuardMatrices}_R(1^m, 1^T, 1^L, 1^N)$:

1. Set constants $c_1 \gg 5(T+2L)$ and $c_2 = 5(T+2L)$. We assume w.l.o.g. that c_1 and c_2 are even.
2. Construct $\mathbf{L}_1 \in R^{5m \times 5m(T+2L)}$ and $\mathbf{R}_{L+N} \in R^{5m(T+2L)\times 5m}$ as follows:

$$\mathbf{L}_1 = \begin{bmatrix} \mathbf{I} & \mathbf{I} & \dots & \mathbf{I} \end{bmatrix}, \quad \mathbf{R}_{L+N} = \begin{bmatrix} \mathbf{I} \\ \mathbf{I} \\ \vdots \\ \mathbf{I} \end{bmatrix},$$

   where $\mathbf{I} \in R^{5m \times 5m}$ is the identity matrix.
3. For each $\ell \in [1, L+N-1]$, construct the guard matrices $\mathbf{L}_{\ell+1} \in R^{5m \times 5m(T+2L)}$ and $\mathbf{R}_\ell \in R^{5m(T+2L)\times 5m}$ as follows:
   (a) 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

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

   where $\mathbf{D}_\ell$ is a uniformly sampled "block-diagonal" matrix $R^{c_2 m \times c_2 m}$, and $\mathbf{J}$ is an *upper triangular* matrix in $R^{c_2 m \times c_2 m}$ with an efficiently computable inverse. More specifically

$$\mathbf{J} = \begin{bmatrix} 1_R & 1_R & 1_R & \dots & 1_R \\ & 1_R & 1_R & \dots & 1_R \\ & & \ddots & \ddots & \vdots \\ & \mathbf{0}_R & & \ddots & 1_R \\ & & & & 1_R \end{bmatrix}$$

   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.
   (b) Sample a uniform square matrix $\mathbf{X}_\ell \leftarrow R^{(c_1/2)m \times (c_1/2)m}$ and a uniform matrix $\mathbf{U}_\ell \leftarrow R^{c_2 m \times (c_1/2)m}$, and set $\mathbf{R}_\ell$ and $\mathbf{L}_{\ell+1}$ as:

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

4. Output $\{(\mathbf{L}_\ell, \mathbf{R}_\ell)\}_{\ell \in [L+N]}$.

Figure 9: Algorithm to Generate Guard Matrices

3. Choose a particular non-diagonal matrix position $(i^*, j^*) \in [5] \times [5]$ [1]. If $\mathbf{Q}_{i^*,j^*} \neq \mathbf{0}_R$, output 0. Otherwise, output 1.

4.5 Proof of Correctness

The following theorem captures the correctness of our iaiO construction $(\mathsf{Create}, \mathsf{Eval})$.

[1] As we explain in Section 2.2 when discussing Barrington's theorem (Theorem 2.1), the final output matrix Q is either the identity matrix or a "particular" permutation matrix that is different from the identity matrix, so we know exactly which single coordinates are viable ones to check when evaluating the obfuscated program.

Theorem 4.2. *For any security parameter λ, any activation matrix $\mathbf{G} \in \{0,1\}^{N \times T \times 2}$,any set of oblivious branching programs $(P_1, \ldots, P_T)$ such that for each $t \in [T]$, P_t is an oblivious branching program of depth L of the form $P_t = \{M_{t,\ell,0}, M_{t,\ell,1}\}_{\ell \in [L]}$, any mapping $\phi : [L] \to [N]$, and any input vector $\mathbf{x} = (x_1, \ldots, x_N) \in \{0,1\}^N$, let the set $S_{\mathbf{x}}$ be defined as:*

$$S_{\mathbf{x}} = \{t \in [T] : \mathbf{G}_{i,t,x_i} = 1 \quad \forall i \in [N]\}.$$

Let $\mathsf{P} \leftarrow \mathsf{Create}(1^\lambda, P = \{P_1, \ldots, P_T\}, \mathbf{G}, \phi)$. Then, whenever $S_{\mathbf{x}} \neq \emptyset$ and $P_t = P_{t'}$ for all $t, t' \in S_{\mathbf{x}}$, the following holds for each $t \in S_{\mathbf{x}}$:

$$\mathsf{Eval}(\mathsf{P}, \mathbf{x}, \phi) = P_t(\mathbf{x}).$$

Proof. We first observe the following:

$$\begin{aligned}
\mathbf{Q} &= \left(\prod_{\ell=1}^{L} \widetilde{\mathbf{P}}_{\ell, x_{\phi(\ell)}}\right) \left(\prod_{i=1}^{N} \widetilde{\mathbf{E}}_{L+i, x_i}\right) \\
&= \mathbf{L}_1 \mathbf{P}_{1, x_{\phi(1)}} \left(\prod_{\ell=2}^{L} \mathbf{D}_{\ell-1} \mathbf{P}_{\ell, x_{\phi(\ell)}}\right) \left(\prod_{i=1}^{N} \mathbf{D}_{L+i-1} \mathbf{E}_{L+i, x_i}\right) \mathbf{R}_{L+N}
\end{aligned}$$

where each $\mathbf{D}_i = \mathbf{R}_i \mathbf{L}_{i+1}$ is a block-diagonal matrix of dimension $5m(T+2L) \times 5m(T+2L)$ for each $i \in [2, L+N-1]$. This follows immediately from the manner in which the guard matrices are generated in Algorithm 9.

We next observe that the consistency-check sub-matrices in each program-carrying matrix $\mathbf{P}_{\ell, x_{\phi(\ell)}}$ and each enforcer matrix $\mathbf{E}_{L+i, x_i}$ do not contribute to the result. To see this, observe the following:

1. For each $(\ell, b) \in [L] \times \{0,1\}$, if $b \neq x_\phi(\ell)$, then the consistency sub-matrices in the corresponding diagonal slot do not contribute to the final product since there is a zero sub-matrix in this diagonal slot in the program carrying matrix $\mathbf{P}_{\ell, x_{\phi(\ell)}}$.
2. For each $(\ell, b) \in [L] \times \{0,1\}$, if $b = x_\phi(\ell)$, then the consistency sub-matrices in the corresponding diagonal slot do not contribute to the final product since there is a zero sub-matrix in this diagonal slot in the enforcer matrix $\mathbf{E}_{L+\phi(\ell), x_{\phi(\ell)}}$.

Hence, we must have

$$\mathbf{Q} = \mathbf{L}_1 \begin{bmatrix} \mathbf{A}_{1, x_{\phi(1)}} \left(\prod_{\ell=2}^{L} \mathbf{D}'_{\ell-1} \mathbf{A}_{\ell, x_{\phi(\ell)}}\right) \left(\prod_{i=1}^{N} \mathbf{D}'_{L+i-1} \mathbf{T}_{L+i, x_i}\right) & & \\ & \mathbf{0} & \\ & & \ddots \\ & & \mathbf{0} \end{bmatrix} \mathbf{R}_{L+N}$$

where each $\mathbf{D}'_i \in R^{5mT \times 5mT}$ is a block-diagonal matrix for each $i \in [2, L+N-1]$ (essentially, each $\mathbf{D}'_i$ is the top left block-diagonal sub-matrix of the corresponding block-diagonal matrix $\mathbf{D}_i \in R^{5m(T+2L) \times 5m((T+2L))}$ above).

Now, for each $\ell \in [L]$, each $i \in [N]$, let

$$\mathbf{A}_{\ell, x_{\phi(\ell)}} = \begin{bmatrix} \mathbf{N}_{\ell,1,x_{\phi(\ell)}} & & \\ & \ddots & \\ & & \mathbf{N}_{\ell,T,x_{\phi(\ell)}} \end{bmatrix}, \quad \mathbf{T}_{L+i,x_i} = \begin{bmatrix} \mathbf{T}_{L+i,1,x_i} & & \\ & \ddots & \\ & & \mathbf{T}_{L+i,T,x_i} \end{bmatrix},$$

Further, for each $j \in [L+N]$, let

$$\mathbf{D}'_j = \begin{bmatrix} \mathbf{D}'_{j,1} & & \\ & \ddots & \\ & & \mathbf{D}'_{j,T} \end{bmatrix}.$$

We further recall that

$$\mathbf{L}_1 = \begin{bmatrix} \mathbf{I} & \mathbf{I} & \dots & \mathbf{I} \end{bmatrix}, \quad \mathbf{R}_{L+N} = \begin{bmatrix} \mathbf{I} \\ \mathbf{I} \\ \vdots \\ \mathbf{I} \end{bmatrix},$$

where $\mathbf{I} \in R^{5m \times 5m}$ is the identity matrix. Hence, we have

$$\begin{aligned} \mathbf{Q} &= \mathbf{L}_1 \begin{bmatrix} \mathbf{A}_{1,x_{\phi(1)}} \left(\prod_{\ell=2}^{L} \mathbf{D}'_{\ell-1} \mathbf{A}_{\ell,x_{\phi(\ell)}} \right) \left(\prod_{i=1}^{N} \mathbf{D}'_{L+i-1} \mathbf{T}_{L+i,x_i} \right) & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & \end{bmatrix} \mathbf{R}_{L+N} \\ &= \sum_{t=1}^{T} \left(\mathbf{A}_{1,t,x_{\phi(1)}} \left(\prod_{\ell=2}^{L} \mathbf{D}'_{\ell,t} \mathbf{A}_{\ell,t,x_{\phi(\ell)}} \right) \left(\prod_{i=1}^{N} \mathbf{D}'_{L+i,t} \mathbf{T}_{L+i,t,x_i} \right) \right). \end{aligned}$$

At this point, 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 $\mathbf{T}$ sub-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 $\mathbf{T}$ sub-matrices will prevent the program P_t from contributing to the final output, and hence the resulting contribution will be 0 in the tested entry if and only if the branching program P_t outputs 0 on the input $\mathbf{x}$. □

This concludes the proof of correctness.

5 Security from Subspace Hiding Assumptions

In this section we define our subspace hiding assumptions, which we use to prove the security of our iaiO construction. These assumptions can be viewed as analogous to the assumption of [GLSW15] in spirit, although syntactically they are very different since they work over matrices of rings rather than in the exponent of a multilinear map (which is a field, and thus much simpler). Later we will present a "core" subspace hiding assumption that is even more similar to the assumption of [GLSW15], but this assumption is less directly related to the proof of iaiO than the assumptions we present here. Looking ahead, we prove that all of these assumptions hold over the output ring of any RHS/RKHwPRF. Before formally describing our assumptions, we introduce certain notations.

5.1 Notations

Diagonal Matrices. Consider a class of block diagonal matrices $\mathbf{D} \in R^{(5mL)\times(5mL)}$. 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 $5m \times 5m$, 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

$$\mathbf{D} = \begin{bmatrix} \mathbf{D}_1 & & & & \\ & \mathbf{D}_2 & & \mathbf{0}_R & \\ & & \mathbf{D}_3 & & \\ & & & \ddots & \\ & \mathbf{0}_R & & & \ddots \\ & & & & \mathbf{D}_L \end{bmatrix},$$

Special Classes of Diagonal Matrices. We introduce some more notation for certain special classes of diagonal matrices which will allow us to simplify our presentation in this section:

1. $\mathbf{D}^i \in R^{(5mL)\times(5mL)}$ denotes a block diagonal matrix with the restriction that *all diagonal blocks except the* i^{th} *diagonal block* are zero, while the ith diagonal block is a uniformly random matrix in $R^{5m\times 5m}$.
2. $\mathbf{D}^{i,j} \in R^{(5mL)\times(5mL)}$ denotes a block diagonal matrix with the restriction that *all diagonal blocks except the* i^{th} *and* j^{th} *diagonal blocks* are zero, while the ith and j^{th} diagonal blocks are uniformly random matrices in $R^{5m\times 5m}$.

Permutation-Embedded Diagonal Matrices. In the aforementioned description, we only considered block-diagonal matrices $\mathbf{D} \in R^{(5mL)\times(5mL)}$ such that each diagonal-block is either zero or random. We also consider some other "more structured" classes of block-diagonal matrices such that one or more of the diagonal blocks (in $R^{5m\times 5m}$) are allowed to be R-embeddings of (arbitrary) permutation matrices in $\{0,1\}^{5\times 5}$. Recall that given a permutation matrix $P \in \{0,1\}^{5\times 5}$, its R-embedding $\mathbf{P}$ is defined as a $5m \times 5m$ matrix over the ring R of the form:

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

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

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

To simplify representation, we introduce some additional notation for permutation-embedded block-diagonal matrices:

1. $\mathbf{D}^{(i,P)} \in R^{(5mL)\times(5mL)}$ denotes a block diagonal matrix with the restriction that *all diagonal blocks except the* i^{th} *diagonal block* are zero, while the ith diagonal block is a $5m \times 5m$ R-embedding of the permutation matrix $P \in \{0,1\}^{5\times 5}$.
2. $\mathbf{D}^{(i,P_1),(j,P_2)} \in R^{(5mL)\times(5mL)}$ denotes a block diagonal matrix with the restriction that *all diagonal blocks except the* i^{th} *and* j^{th} *diagonal blocks* are zero, while the ith and j^{th} diagonal blocks are $5m \times 5m$ R-embeddings of the permutation matrices $P_1, P_2 \in \{0,1\}^{5\times 5}$, respectively.
3. $\mathbf{D}^{(i,P),j} \in R^{(5mL)\times(5mL)}$ denotes a block diagonal matrix with the restriction that *all diagonal blocks except the* i^{th} *and* j^{th} *diagonal blocks* are zero, while the ith and j^{th} diagonal blocks are $5m \times 5m$ matrices of the following form:
   - The ith diagonal block is a $5m \times 5m$ R-embedding of the permutation matrix $P \in \{0,1\}^{5\times 5}$.
   - The jth diagonal block is a uniformly random matrix in $R^{5m\times 5m}$.

The Guard Matrices. Let n, m and L be integers such that $5mL \ll n$. We use a set of matrices $\mathbf{L}_i \in R^{n \times (5mL)}$ for $i \in [2, N]$, and $\mathbf{R}_i \in R^{(5mL) \times n}$ for $i \in [2, N]$. In addition, we consider sub-matrices of $\mathbf{L}_i$ and $\mathbf{R}_i$, respectively. Also:

- Let $\mathbf{L}_{i,j} \in R^{n \times 5m}$ be the sub-matrix of $\mathbf{L}_i$ consisting of the $(5m\,(j-1)+1)$-th through the $5mj$-th *columns* of $\mathbf{L}_i$.
- Let $\mathbf{R}_{i,j} \in R^{5m \times n}$ be the sub-matrix of $\mathbf{R}_i$ consisting of the $(5m\,(j-1)+1)$-th through the $5mj$-th *rows* of $\mathbf{R}_i$.

In other words, our matrices have the following structure:

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

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

$$\mathbf{R}_{i,j} \cdot \mathbf{L}_{i+1,j'} = \mathbf{0}_R \in R^{5m \times 5m}.$$

In other words, the product $\mathbf{R}_i \cdot \mathbf{L}_{i+1} \in R^{5mL \times 5mL}$ is required to have the form of a block diagonal matrix $\mathbf{D}$ as we defined above. Finally, the "end guards" $\mathbf{L}_1$ and $\mathbf{R}_N$ are specially structured, as described in Section 4.3, i.e. we have

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

5.2 The Subspace Hiding Assumptions

Equipped with the aforementioned notations, we now state the subspace hiding assumption that we require to prove security of our iaiO construction. For ease of exposition, we introduce two versions of this assumption, each of which allows to prove certain properties of our iaiO scheme.

Definition 5.1. (Subspace Hiding Assumption-1.) For any ring R, any choice of parameters m, L, N, T, Z and each $b \in \{0,1\}$, let $\mathsf{Expt}^{\text{Subspace-Hiding-1}}_{R,m,L,N,T,Z,b}$ denote an experiment between a challenger and an adversary $\mathcal{A}$ as defined in Figure 10. The subspace hiding assumption-1 is said to hold over a ring R if for any security parameter λ, any choice of parameters $m, L, N, T, Z = \mathrm{poly}(\lambda)$ and for any PPT adversary $\mathcal{A}$, the views of the adversary $\mathcal{A}$ in the experiments $\mathsf{Expt}^{\text{Subspace-Hiding-1}}_{R,m,L,N,T,Z,0}$ and $\mathsf{Expt}^{\text{Subspace-Hiding-1}}_{R,m,L,N,T,Z,1}$ are computationally indistinguishable.

Definition 5.2. (Subspace Hiding Assumption-2.) For any ring R, any choice of parameters L, N, T, Z and each $b \in \{0,1\}$, let $\mathsf{Expt}^{\text{Subspace-Hiding-2}}_{R,m,L,N,T,Z,b}$ denote an experiment between a challenger and an adversary $\mathcal{A}$ as defined in Figure 11. The subspace hiding assumption-2 is said to hold over a ring R if for any security parameter λ, any choice of parameters $L, N, T, Z = \mathrm{poly}(\lambda)$ and for any PPT adversary $\mathcal{A}$, the views of the adversary $\mathcal{A}$ in the experiments $\mathsf{Expt}^{\text{Subspace-Hiding-2}}_{R,m,L,N,T,Z,0}$ and $\mathsf{Expt}^{\text{Subspace-Hiding-2}}_{R,m,L,N,T,Z,1}$ are computationally indistinguishable.

Looking ahead, the subspace hiding assumption-1 allows us to prove inter-column security and intra-column security of our iaiO construction, while subspace hiding assumption-2 allows us to prove that our iaiO construction achieves completely inactive program security and single-input program switching security. Observe that the two assumptions

Experiment $\mathsf{Expt}^{\text{Subspace-Hiding-1}}_{R,m,L,N,T,Z,b}$:

1. The adversary $\mathcal{A}$ chooses two challenge permutation-slots $t^*_0, t^*_1 \in [T]$, a challenge-index $i^* \in [N]$, and T programs $\{P_t\}_{t\in[T]}$, where for each $t \in [T]$, we have $P_t = \{N_{t,\ell,b}\}_{\ell\in[L],b\in\{0,1\}}$, subject to the restriction that $P_{t^*_0} = P_{t^*_1}$. It provides $(t^*_0, t^*_1, i^*, \{P_t\}_{t\in[T]})$ to the challenger.
2. The challenger provides the adversary with the following matrices sampled from appropriate distributions (in what follows, all block diagonal matrices $\mathbf{D}$ are of dimension $5m(T+Z) \times 5m(T+Z)$, and a total of $(L+N)$ guard matrix pairs are used, where each guard-matrix pair is distributed according to the distribution mentioned earlier):

   (a) **Generators:**

   - For each $\ell \in [L]$, $b \in \{0,1\}$ and each $t \in [T] \setminus \{t^*_1\}$: polynomially many samples of the form $\mathbf{L}_\ell \mathbf{D}^{(t,N_{t,\ell,b})} \mathbf{R}_\ell$, where $\mathbf{D}^{(t,N_{t,\ell,b})}$ is a permutation-embedded block-diagonal matrix sampled from the appropriate distribution.
   - For each $i \in [N]$ and each $t \in [T] \setminus \{t^*_1\}$: polynomially many samples of the form $\mathbf{L}_{L+i} \mathbf{D}^{(t,I)} \mathbf{R}_{L+i}$, where $I \in \{0,1\}^{5\times 5}$ denotes the identity permutation, and $\mathbf{D}^{(t,I)}$ is an identity-permutation-embedded block-diagonal matrix sampled from the appropriate distribution.
   - For each $\ell \in [L+N]$ and each $z \in [Z]$: polynomially many samples of the form $\mathbf{L}_\ell \mathbf{D}^{(T+z)} \mathbf{R}_\ell$, where $\mathbf{D}^{(T+z)}$ is a block-diagonal matrix sampled from the appropriate distribution.

   In each case, the "end guards" $\mathbf{L}_1$ and $\mathbf{R}_{L+N}$ are specially structured, as described in Section 4.3.

   (b) **Challenge-Relevant Elements**:

   - For each $\ell \in [L]$ and for each $b \in \{0,1\}$: polynomially many samples of the form $\mathbf{L}_\ell \mathbf{D}^{(t^*_0,N_{t^*_0,\ell,b}),(t^*_1,N_{t^*_1,\ell,b})} \mathbf{R}_\ell$, where $\mathbf{D}^{(t^*_0,N_{t^*_0,\ell,b}),(t^*_1,N_{t^*_1,\ell,b})}$ is a permutation-embedded block-diagonal matrix sampled from the appropriate distribution.
   - For each $i \in [N]$: polynomially many samples of the form $\mathbf{L}_{L+i} \mathbf{D}^{(t^*_0,I),(t^*_1,I)} \mathbf{R}_{L+i}$, where $I \in \{0,1\}^{5\times 5}$ denotes the identity permutation, and $\mathbf{D}^{(t^*_0,I),(t^*_1,I)}$ is an identity-permutation-embedded block-diagonal matrix sampled from the appropriate distribution.

   Once again, in each case, the "end guards" $\mathbf{L}_1$ and $\mathbf{R}_{L+N}$ are specially structured, as described in Section 4.3.

   (c) **Special Challenge-Index Elements**: Polynomially many samples of the form $\mathbf{L}_{(L+i^*)} \mathbf{D}^{(t^*_1,I)} \mathbf{R}_{(L+i^*)}$, where $I \in \{0,1\}^{5\times 5}$ again denotes the identity permutation, and $\mathbf{D}^{(t^*_1,I)}$ is an identity-permutation-embedded block-diagonal matrix sampled from the appropriate distribution.

   Also, in the case when $i^* = N$, the right "end guard" $\mathbf{R}_{L+N}$ is specially structured, as described in Section 4.3.

   (d) **The Challenge Element**: A sample that has one of the two following forms:

   - $\mathbf{L}_{(L+i^*)} \mathbf{D}^{(t^*_0,I)} \mathbf{R}_{(L+i^*)}$ if $b = 0$, or
   - $\mathbf{L}_{(L+i^*)} \mathbf{D}^{(t^*_0,I),(t^*_1,I)} \mathbf{R}_{(L+i^*)}$ if $b = 1$,

   where $I \in \{0,1\}^{5\times 5}$ again denotes the identity permutation, and $\mathbf{D}^{(t^*_0,I)}$ and $\mathbf{D}^{(t^*_0,I),(t^*_1,I)}$ are block-diagonal matrices sampled from the appropriate distribution. Also, in the case when $i^* = N$, the right "end guard" $\mathbf{R}_{L+N}$ is specially structured, as described in Section 4.3.

Figure 10: The Subspace Hiding Assumption-1

Experiment $\mathsf{Expt}^{\text{Subspace-Hiding-2}}_{R,m,L,N,T,Z,b}$:

1. The adversary $\mathcal{A}$ chooses a challenge permutation-slot $t^* \in [T]$, an additional challenge slot $z^* \in [Z]$, a challenge-index $\ell^* \in [L]$, a challenge-bit $\beta \in \{0,1\}$ and T programs $\{P_t\}_{t\in[T]}$, where for each $t \in [T]$, we have $P_t = \{N_{t,\ell,b}\}_{\ell\in[L],b\in\{0,1\}}$. It provides $(t^*, z^*, \ell^*, \beta, \{P_t\}_{t\in[T]})$ to the challenger.
2. The challenger provides the adversary with the following matrices sampled from appropriate distributions (in what follows, all block diagonal matrices $\mathbf{D}$ are of dimension $5m(T+Z) \times 5m(T+Z)$, and a total of $(L+N)$ guard matrix pairs are used, where each guard-matrix pair is distributed according to the distribution mentioned earlier):

   (a) **Generators:**

   - For each $\ell \in [L]$, $b \in \{0,1\}$ and each $t \in [T] \setminus \{t^*\}$: polynomially many samples of the form $\mathbf{L}_\ell \mathbf{D}^{(t,N_{t,\ell,b})} \mathbf{R}_\ell$, where $\mathbf{D}^{(t,N_{t,\ell,b})}$ is a permutation-embedded block-diagonal matrix sampled from the appropriate distribution.
   - For each $i \in [N]$ and each $t \in [T] \setminus \{t^*\}$: polynomially many samples of the form $\mathbf{L}_{L+i} \mathbf{D}^{(t,I)} \mathbf{R}_{L+i}$, where $I \in \{0,1\}^{5\times 5}$ denotes the identity permutation, and $\mathbf{D}^{(t,I)}$ is an identity-permutation-embedded block-diagonal matrix sampled from the appropriate distribution.
   - For each $\ell \in [L+N]$ and each $z \in [Z]$: polynomially many samples of the form $\mathbf{L}_\ell \mathbf{D}^{(T+z)} \mathbf{R}_\ell$, where $\mathbf{D}^{(T+z)}$ is a block-diagonal matrix sampled from the appropriate distribution.

   In each case, the "end guards" $\mathbf{L}_1$ and $\mathbf{R}_{L+N}$ are specially structured, as described in Section 4.3.

   (b) **Challenge-Relevant Elements**:

   - For each $\ell \in [L]$ and $b \in \{0,1\}$: polynomially many samples of the form $\mathbf{L}_\ell \mathbf{D}^{(t^*,N_{t^*,\ell,b}),z^*} \mathbf{R}_\ell$, where $\mathbf{D}^{(t^*,N_{t^*,\ell,b}),z^*}$ is a permutation-embedded block-diagonal matrix sampled from the appropriate distribution, with an R-embedding of the permutation matrix $N_{t^*,\ell,b}$ in the t^*-th diagonal block and a uniform sub-matrix in the z^*-th diagonal block.
   - For each $i \in [N]$: polynomially many samples of the form $\mathbf{L}_{L+i} \mathbf{D}^{(t^*,I),z^*} \mathbf{R}_{L+i}$, where $I \in \{0,1\}^{5\times 5}$ denotes the identity permutation, and $\mathbf{D}^{(t^*,I),z^*}$ is an identity permutation-embedded block-diagonal matrix sampled from the appropriate distribution, with an R-embedding of I in the t^*-th diagonal block and a uniform sub-matrix in the z^*-th diagonal block.

   Once again, in each case, the "end guards" $\mathbf{L}_1$ and $\mathbf{R}_{L+N}$ are specially structured, as described in Section 4.3.

   (c) **Special Challenge-Index Elements**: polynomially many samples of the form $\mathbf{L}_{\ell^*} \mathbf{D}^{(t^*,N_{t^*,\ell^*,\beta})} \mathbf{R}_{\ell^*}$, where $\mathbf{D}^{(t^*,P_{N^*,\ell^*,\beta})}$ is a permutation-embedded block-diagonal matrix sampled from the appropriate distribution.

   Also, in the case when $\ell^* = 1$, the left "end guard" $\mathbf{L}_1$ is specially structured, as described in Section 4.3.

   (d) **The Challenge Element**: A sample that has one of the two following forms:

   - $\mathbf{L}_{\ell^*} \mathbf{D}^{z^*} \mathbf{R}_{\ell^*}$ if $b = 0$, or
   - $\mathbf{L}_{\ell^*} \mathbf{D}^{(t^*,N_{t^*,\ell^*,\beta}),z^*} \mathbf{R}_{\ell^*}$ if $b = 1$,

   where:

   - $\mathbf{D}^{z^*}$ is a block-diagonal matrix sampled from the appropriate distribution, with a uniform sub-matrix in the z^*-th diagonal block.
   - $\mathbf{D}^{(t^*,N_{t^*,\ell,\beta}),z^*}$ is a permutation-embedded block-diagonal matrix sampled from the appropriate distribution, with an R-embedding of the permutation matrix $N_{t^*,\ell^*,\beta}$ in the t^*-th diagonal block and a uniform sub-matrix in the z^*-th diagonal block.

   Also, in the case when $\ell^* = 1$, the left "end guard" $\mathbf{L}_1$ is specially structured, as described in Section 4.3.

Figure 11: The Subspace Hiding Assumption-2

differ primarily in the nature of the challenge terms given to the adversary, while the remaining terms given to the adversary are mostly similar.

We informally discuss why the subspace hiding assumptions are designed in this manner. Intuitively, the challenge terms in the subspace hiding assumption-1 are so designed to allow switching the entries of the input-activation matrix $\mathbf{G}$ in an iaiO instance (by changing the corresponding entries in the enforcer sub-matrices used to encode the input-activation information in our iaiO construction) in an indistinguishable manner. This is, broadly speaking, what the challenger does in both the inter-column and intra-column security games. On the other hand, the challenge terms in the subspace hiding assumption-2 are so designed to allow switching the input programs $(P_1, \ldots, P_T)$ in an iaiO instance (by changing the corresponding entries in the program-carrying sub-matrices used to encode the programs in our iaiO construction). Again, broadly speaking, this is what the challenger does in both the completely inactive program security game and single-input program switching security game. The remaining terms given to the adversary in both assumptions are primarily used to faithfully simulate the remaining parts of the obfuscated program in each of the security games.

The actual security proofs of our iaiO construction based on the above assumptions (detailed in Section 6) are significantly more nuanced, but mostly follow the above intuition.

The Baby Subspace Hiding Assumptions. We also define special cases of the aforementioned assumptions, called the "baby" subspace hiding assumptions.

- The baby subspace hiding assumption-1 is a special case of the subspace hiding assumption-1 where there are only two permutation slots (which are, by default, the challenge permutation slots) and no additional slots, i.e., we have $T = 2$ and $Z = 0$. We (informally) refer to this baby subspace-hiding assumption as the "permutation" baby subspace-hiding assumption.
- The baby subspace hiding assumption-2 is a special case of the subspace hiding assumption-2 where there is only one permutation slot and one additional slot (which are, by default, the challenge permutation slot and the additional challenge slot), i.e., we have $T = 1$ and $Z = 1$. We (informally) refer to this baby subspace-hiding assumption as the "half-permutation" baby subspace-hiding assumption.

For the sake of completeness, we describe these assumptions below.

Definition 5.3. (Baby Subspace Hiding Assumption-1.) The baby subspace hiding assumption-1 is said to hold over a ring R if for any security parameter λ, any choice of parameters $m, L, N = \mathrm{poly}(\lambda)$ and for any PPT adversary $\mathcal{A}$, the views of the adversary $\mathcal{A}$ in the experiments $\mathsf{Expt}^{\text{Subspace-Hiding-1}}_{R,m,L,N,2,0,0}$ and $\mathsf{Expt}^{\text{Subspace-Hiding-1}}_{R,m,L,N,2,0,1}$ are computationally indistinguishable.

Definition 5.4. (Baby Subspace Hiding Assumption-2.) The baby subspace hiding assumption-2 is said to hold over a ring R if for any security parameter λ, any choice of parameters $m, L, N = \mathrm{poly}(\lambda)$ and for any PPT adversary $\mathcal{A}$, the views of the adversary $\mathcal{A}$ in the experiments $\mathsf{Expt}^{\text{Subspace-Hiding-2}}_{R,m,L,N,1,1,0}$ and $\mathsf{Expt}^{\text{Subspace-Hiding-2}}_{R,m,L,N,1,1,1}$ are computationally indistinguishable.

5.3 The Main Theorems

We prove the following main theorems.

Theorem 5.5. *Assuming that the subspace hiding assumptions 1 and 2 as described in Definitions 5.1 and 5.2 hold, our iaiO construction achieves inter-column security, single-input program switching security, completely inactive program security, and intra-column security, as in Definition 3.7 (Section 3.2.4).*

Theorem 5.6. *Let R be any generic ring such that for any $m, Q = \mathrm{poly}(\lambda)$ where m is any arbitrary positive integer satisfying $m > 3\lambda$ and Q is any arbitrary positive integer, we have*

$$(\mathbf{A}, \mathbf{AK}) \overset{c}{\approx} (\mathbf{A}, \mathbf{U}), \quad (\mathbf{B}, \mathbf{K}'\mathbf{B}) \overset{c}{\approx} (\mathbf{B}, \mathbf{V})$$

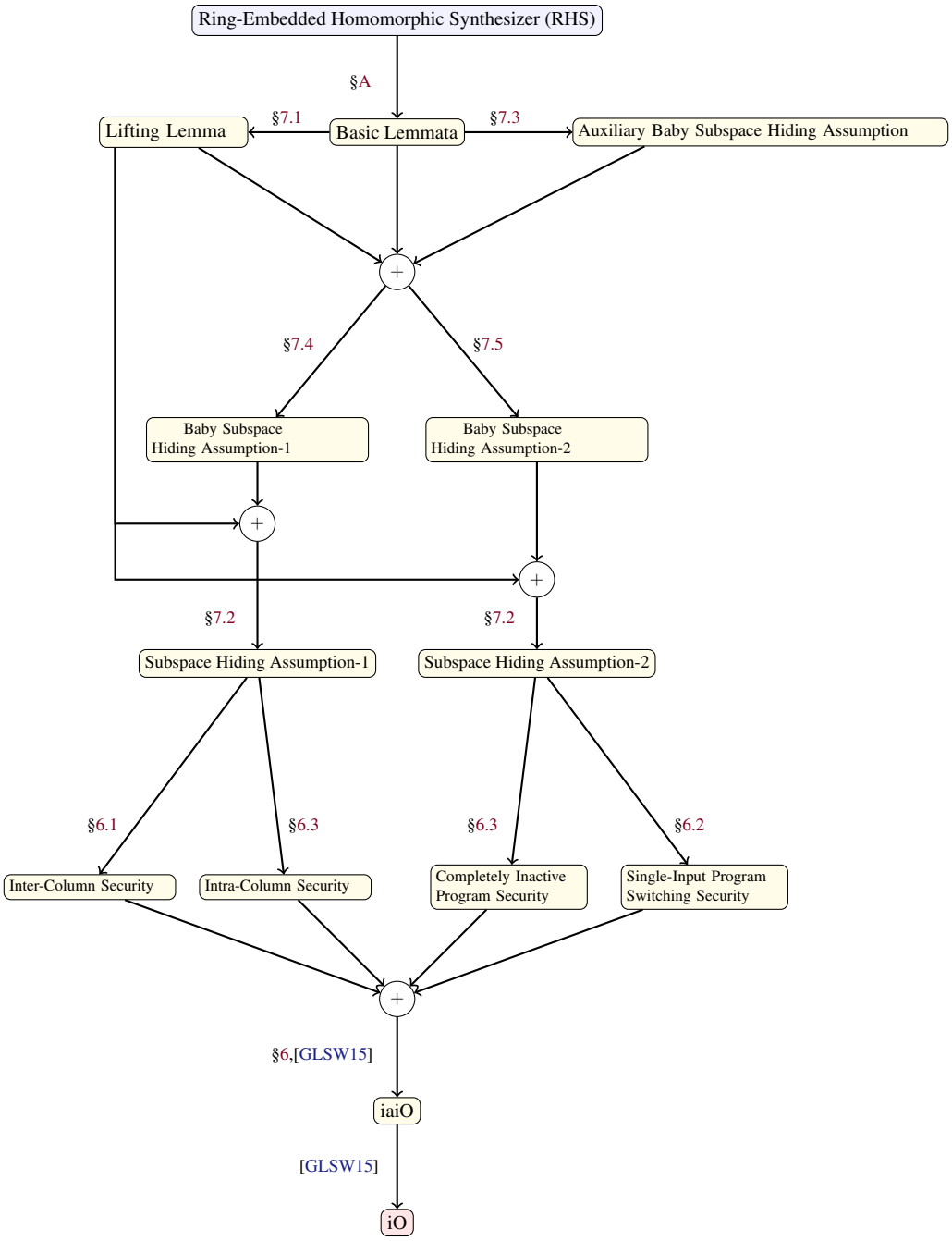

Figure 12: Organization of the Proofs of Security for Our iaiO Construction

where $\mathbf{K}, \mathbf{K}' \leftarrow R^{m \times m}$, $\mathbf{A}, \mathbf{U} \leftarrow R^{Qm \times m}$, *and* $\mathbf{B}, \mathbf{V} \leftarrow R^{m \times Qm}$. *Then the subspace hiding assumptions 1 and 2 as described in Definitions 5.1 and 5.2 hold over the ring* R.

Theorem 5.7. *Let* R *be the output space of any (generic) ring-embedded homomorphic synthesizer (RHS) as in Definition 3.3 (Section 3.2.2), with appropriately chosen parameters such that Theorems 3.11 and 3.13 hold over* R. *Then the subspace hiding assumptions 1 and 2 as described in Definitions 5.1 and 5.2 hold over the ring* R.

Note that the proof of Theorem 5.7 follows immediately from Theorem 5.6 and Corollary 3.15. Put together, these theorems show that the existence of any generic ring-embedded homomorphic synthesizer (and hence, by Claim 3.4, the existence of any generic RKHwPRF) implies the existence of a secure iaiO scheme, captured in the following main theorem and corollary.

Theorem 5.8 (Main Theorem). *Assuming the existence of any generic RHS, there exists a secure input-activated iO scheme as defined in [GLSW15] for all circuits in $\mathcal{NC}^1$.*

Corollary 5.9 (Main Corollary). *Assuming the existence of any generic RKHwPRF, there exists a secure input-activated iO scheme as defined in [GLSW15] for all circuits in $\mathcal{NC}^1$.*

Proof Organization. The proofs of the above main theorems are presented in the supplementary material. Concretely, we prove Theorem 5.5 in Section 6 and Theorem 5.6 in Section 7. The organization of these proofs is depicted pictorially in Figure 12. We hope that this provides the reader with a high-level roadmap for how the overall proof of security of our iaiO construction from any RHS is broken up into various smaller steps, and how these steps are arranged across the various sections in the rest of the paper.

6 Proof of Theorem 5.5

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 assumptions as defined in Section 5. Refer Section 3.2.4 for the formal definitions of these security notions for any iaiO scheme.

6.1 Inter-Column Security

Lemma 6.1. *Assuming that the subspace hiding assumption-1 as described in Definition 5.1 holds, our iaiO construction achieves inter-column security as in Definition 3.7.*

Proof. As a first step of the proof, consider a "modified" inter-column security game where the challenger sets the Kilian randomizer matrices corresponding to the challenge programs to be uniformly random but *identical*. We prove that the view of the adversary in this "modified" inter-column security game is *computationally indistinguishable* from that in the original inter-column security game where the Kilian randomizer matrices corresponding to the challenge programs are uniformly random and independently sampled. In particular, this implies that any PPT adversary $\mathcal{A}$ that wins the inter-column security game with non-negligible advantage also wins the "modified" inter-column security game with non-negligible advantage.

Proof of Transition Step. We now formally prove the step that allows us to transition in a computationally indistinguishable manner from the original inter-column security game where the Kilian randomizer matrices corresponding to the challenge programs are uniformly random and independently sampled, to the "modified" inter-column security game where these randomizer matrices are set to be identical. For simplicity, assume that the challenge program slots chosen by the adversary are $(t_0^*, t_1^*) = (1, 2)$. This is only for ease of exponentiation; our argument generalizes naturally to any arbitrary choice of (t_0^*, t_1^*), as we demonstrate subsequently.

Recall that at any level $\ell \in [L]$ and for any bit $b \in \{0, 1\}$, the aggregate program carrying matrix is structured as

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

Consider the sub-matrix $\mathbf{A}_{\ell,b}^{(1,2)}$ of the form

$$\mathbf{A}_{\ell,b}^{(1,2)} = \begin{bmatrix} \mathbf{N}_{1,\ell,b} & \\ & \mathbf{N}_{2,\ell,b} \end{bmatrix}.$$

In the *original* inter-column security game where the Kilian randomizer matrices corresponding to the challenge programs are uniformly random and independently sampled, we have

$$\mathbf{A}_{\ell,b}^{(1,2)} = \begin{bmatrix} \mathbf{Z}_{1,\ell}\mathbf{M}_{1,\ell,b}\bar{\mathbf{Z}}_{1,\ell} & \\ & \mathbf{Z}_{2,\ell}\mathbf{M}_{2,\ell,b}\bar{\mathbf{Z}}_{2,\ell} \end{bmatrix},$$

where for $t \in \{1,2\}$, $\mathbf{Z}_{t,\ell}$ and $\bar{\mathbf{Z}}_{t,\ell}$ are ring-embedded permutation matrices corresponding to the (plaintext) Kilian randomizer matrices $Z_{t,\ell}$ and $Z_{t,\ell}^{-1}$, respectively. On the other hand, in the "modified" inter-column security game where the Kilian randomizer matrices corresponding to the challenge programs are uniformly identical, we have

$$\widetilde{\mathbf{A}}_{\ell,b}^{(1,2)} = \begin{bmatrix} \widetilde{\mathbf{Z}}_{\ell}\mathbf{M}_{1,\ell,b}\widetilde{\bar{\mathbf{Z}}}_{\ell} & \\ & \widetilde{\mathbf{Z}}_{\ell}\mathbf{M}_{2,\ell,b}\widetilde{\bar{\mathbf{Z}}}_{\ell} \end{bmatrix},$$

where again, $\widetilde{\mathbf{Z}}_{\ell}$ and $\widetilde{\bar{\mathbf{Z}}}_{\ell}$ are ring-embedded permutation matrices corresponding to the (plaintext) Kilian randomizer matrices $\widetilde{Z}_{\ell}$ and $\widetilde{Z}_{\ell}^{-1}$, respectively. Note that apart from this, the remainder of the aggregate program-carrying sub-matrices (corresponding to the non-challenge program slots), the consistency check sub-matrices and the enforcer matrices are distributed identically in both games.

Now, observe that in the original and modified inter-column security games, the adversary $\mathcal{A}$ does not observe the sub-matrices $\mathbf{A}_{\ell,b}^{(1,2)}$ or $\widetilde{\mathbf{A}}_{\ell,b}^{(1,2)}$ in the clear, since it does not observe the program-carrying matrix $\mathbf{P}_{\ell,b}$ in the clear. Instead, it observes "guarded" versions of these sub-matrices as sub-matrices of the guarded program-carrying matrix $\widetilde{\mathbf{P}}_{\ell,b}$, where

$$\widetilde{\mathbf{P}}_{\ell,b} = \mathbf{L}_{\ell}\mathbf{P}_{\ell,b}\mathbf{R}_{\ell}.$$

More specifically, the adversary $\mathcal{A}$ in the *original* inter-column security game observes the sub-matrix

$$\mathbf{L}_{\ell}^{(1,2)}\mathbf{A}_{\ell,b}^{(1,2)}\mathbf{R}_{\ell}^{(1,2)} = \begin{bmatrix} \mathbf{L}_{\ell}^{(1)}\mathbf{Z}_{1,\ell}\mathbf{M}_{1,\ell,b}\bar{\mathbf{Z}}_{1,\ell}\mathbf{R}_{\ell}^{(1)} & \\ & \mathbf{L}_{\ell}^{(2)}\mathbf{Z}_{2,\ell}\mathbf{M}_{2,\ell,b}\bar{\mathbf{Z}}_{2,\ell}\mathbf{R}_{\ell}^{(2)} \end{bmatrix},$$

where:

- $\mathbf{L}_{\ell}^{(1,2)}$ is the sub-matrix consisting of the first two $5m$-blockwise columns of the guard matrix $\mathbf{L}_{\ell}$.
- $\mathbf{L}_{\ell}^{(1)}$ (resp. $\mathbf{L}_{\ell}^{(2)}$) is the sub-matrix consisting of the first (resp., second) $5m$-blockwise column of the guard matrix $\mathbf{L}_{\ell}$.
- $\mathbf{R}_{\ell}^{(1,2)}$ is the sub-matrix consisting of the first two $5m$-blockwise rows of the guard matrix $\mathbf{R}_{\ell}$.
- $\mathbf{R}_{\ell}^{(1)}$ (resp. $\mathbf{R}_{\ell}^{(2)}$) is the sub-matrix consisting of the first (resp., second) $5m$-blockwise row of the guard matrix $\mathbf{R}_{\ell}$.

Similarly, the adversary $\mathcal{A}$ in the "modified" inter-column security game observes the sub-matrix

$$\mathbf{L}_{\ell}^{(1,2)}\widetilde{\mathbf{A}}_{\ell,b}^{(1,2)}\mathbf{R}_{\ell}^{(1,2)} = \begin{bmatrix} \mathbf{L}_{\ell}^{(1)}\widetilde{\mathbf{Z}}_{\ell}\mathbf{M}_{1,\ell,b}\widetilde{\bar{\mathbf{Z}}}_{\ell}\mathbf{R}_{\ell}^{(1)} & \\ & \mathbf{L}_{\ell}^{(2)}\widetilde{\mathbf{Z}}_{\ell}\mathbf{M}_{2,\ell,b}\widetilde{\bar{\mathbf{Z}}}_{\ell}\mathbf{R}_{\ell}^{(2)} \end{bmatrix}.$$

Now, we claim that

$$\mathbf{L}_{\ell}^{(1,2)}\mathbf{A}_{\ell,b}^{(1,2)}\mathbf{R}_{\ell}^{(1,2)} \quad \overset{c}{\approx} \quad \mathbf{L}_{\ell}^{(1,2)}\widetilde{\mathbf{A}}_{\ell,b}^{(1,2)}\mathbf{R}_{\ell}^{(1,2)}.$$

To begin with, we apply Corollary A.2 (which is proved subsequently in Appendix A) to argue that for each $\ell \in [L+N-1]$, and each $j, j' \in [L+N]$ we have that the sub-matrices $\mathbf{R}_{\ell}^{(j)}$ and $\mathbf{L}_{\ell+1}^{j'}$ are distributed in a manner that is computationally indistinguishable from independent and uniformly random block-diagonal matrices over $R^{5m\times 5m(T+2L)}$ and $R^{5m(T+2L)\times 5m}$, respectively, subject to the restriction that

$$\mathbf{R}_{\ell}^{(j)} \cdot \mathbf{L}_{\ell+1}^{(j')} = \begin{cases} \mathbf{0}_R & \text{if } j \neq j', \\ \text{uniform block-diagonal in } R^{5m\times 5m} & \text{otherwise.} \end{cases}.$$

More formally, Corollary A.2 allows us to argue that

$$(\mathbf{L}_\ell^{(1)}, \mathbf{L}_\ell^{(2)}, \mathbf{R}_\ell^{(1)}, \mathbf{R}_\ell^{(2)}) \overset{c}{\approx} (\widehat{\mathbf{L}}_\ell^{(1)}, \widehat{\mathbf{L}}_\ell^{(2)}, \widehat{\mathbf{R}}_\ell^{(1)}, \widehat{\mathbf{R}}_\ell^{(2)}),$$

where the sub-matrices $\widehat{\mathbf{R}}_\ell^{(1)}$ and $\widehat{\mathbf{L}}_{\ell+1}^{2}$ are independent and uniformly random block-diagonal matrices over $R^{5m\times 5m(T+2L)}$ and $R^{5m(T+2L)\times 5m}$, respectively, subject to the restriction that

$$\widehat{\mathbf{R}}_\ell^{(j)} \cdot \widehat{\mathbf{L}}_{\ell+1}^{(j')} = \begin{cases} \mathbf{0}_R & \text{if } j \neq j', \\ \text{uniform block-diagonal in } R^{5m\times 5m} & \text{otherwise.} \end{cases}.$$

Now, we claim that

$$(\widehat{\mathbf{L}}_\ell^{(1)}, \widehat{\mathbf{L}}_\ell^{(2)}, \widehat{\mathbf{R}}_\ell^{(1)}, \widehat{\mathbf{R}}_\ell^{(2)}) \overset{s}{\approx} (\widehat{\mathbf{L}}_\ell^{(1)}\mathbf{Z}_{1,\ell}, \widehat{\mathbf{L}}_\ell^{(2)}\mathbf{Z}_{2,\ell}, \bar{\mathbf{Z}}_{1,\ell}\widehat{\mathbf{R}}_\ell^{(1)}, \bar{\mathbf{Z}}_{1,\ell}\widehat{\mathbf{R}}_\ell^{(2)}),$$

and also

$$(\widehat{\mathbf{L}}_\ell^{(1)}, \widehat{\mathbf{L}}_\ell^{(2)}, \widehat{\mathbf{R}}_\ell^{(1)}, \widehat{\mathbf{R}}_\ell^{(2)}) \overset{s}{\approx} (\widehat{\mathbf{L}}_\ell^{(1)}\widetilde{\mathbf{Z}}_\ell, \widehat{\mathbf{L}}_\ell^{(2)}\widetilde{\mathbf{Z}}_\ell, \bar{\widetilde{\mathbf{Z}}}_\ell\widehat{\mathbf{R}}_\ell^{(1)}, \bar{\widetilde{\mathbf{Z}}}_\ell\widehat{\mathbf{R}}_\ell^{(2)}).$$

To see this, observe that in both cases, we are essentially right and left multiplying the guard sub-matrices by ring-embedded permutation matrices corresponding to the Kilian randomizer matrices. This has the effect of *permuting the rows and columns of the guard sub-matrices*, respectively. Since the guard sub-matrices are independent and uniformly random subject to the constraints described above, such a permutation does not alter their distribution in any way, irrespective of whether they are permuted using the same or different permutation. Hence, we have

$$(\widehat{\mathbf{L}}_\ell^{(1)}\mathbf{Z}_{1,\ell}, \widehat{\mathbf{L}}_\ell^{(2)}\mathbf{Z}_{2,\ell}, \bar{\mathbf{Z}}_{1,\ell}\widehat{\mathbf{R}}_\ell^{(1)}, \bar{\mathbf{Z}}_{1,\ell}\widehat{\mathbf{R}}_\ell^{(2)}) \overset{s}{\approx} (\widehat{\mathbf{L}}_\ell^{(1)}\widetilde{\mathbf{Z}}_\ell, \widehat{\mathbf{L}}_\ell^{(2)}\widetilde{\mathbf{Z}}_\ell, \bar{\widetilde{\mathbf{Z}}}_\ell\widehat{\mathbf{R}}_\ell^{(1)}, \bar{\widetilde{\mathbf{Z}}}_\ell\widehat{\mathbf{R}}_\ell^{(2)}),$$

which in turn implies that

$$\widehat{\mathbf{L}}_\ell^{(1,2)}\mathbf{A}_{\ell,b}^{(1,2)}\widehat{\mathbf{R}}_\ell^{(1,2)} \quad \overset{s}{\approx} \quad \widehat{\mathbf{L}}_\ell^{(1,2)}\widetilde{\mathbf{A}}_{\ell,b}^{(1,2)}\widehat{\mathbf{R}}_\ell^{(1,2)}.$$

Finally, by a hybrid application of Corollary A.2, we have

$$\begin{aligned} \mathbf{L}_\ell^{(1,2)}\mathbf{A}_{\ell,b}^{(1,2)}\mathbf{R}_\ell^{(1,2)} \quad &\overset{c}{\approx} \quad \widehat{\mathbf{L}}_\ell^{(1,2)}\mathbf{A}_{\ell,b}^{(1,2)}\widehat{\mathbf{R}}_\ell^{(1,2)} \\ &\overset{s}{\approx} \quad \widehat{\mathbf{L}}_\ell^{(1,2)}\widetilde{\mathbf{A}}_{\ell,b}^{(1,2)}\widehat{\mathbf{R}}_\ell^{(1,2)} \\ &\overset{c}{\approx} \quad \mathbf{L}_\ell^{(1,2)}\widetilde{\mathbf{A}}_{\ell,b}^{(1,2)}\mathbf{R}_\ell^{(1,2)}, \end{aligned}$$

where we apply Corollary A.2 in the first and third steps. This, in turn, implies that the guarded program-carrying matrices are computationally indistinguishable in the two games (this follows from the fact that the remainder of the aggregate program-carrying sub-matrices (corresponding to the non-challenge program slots), the consistency check sub-matrices and the enforcer matrices are distributed identically in both games).

Finally, it is easy to see that the argument generalizes in a straightforward manner to any arbitrary pair of challenge slots (t_0^*, t_1^*). In particular, we will use an almost identical argument to show that

$$\mathbf{L}_\ell^{(t_0^*,t_1^*)}\mathbf{A}_{\ell,b}^{(t_0^*,t_1^*)}\mathbf{R}_\ell^{(t_0^*,t_1^*)} \overset{c}{\approx} \mathbf{L}_\ell^{(t_0^*,t_1^*)}\widetilde{\mathbf{A}}_{\ell,b}^{(t_0^*,t_1^*)}\mathbf{R}_\ell^{(t_0^*,t_1^*)},$$

where the corresponding sub-matrices are defined analogously. This completes the first step of the proof that allows us to transition in a computationally indistinguishable manner from the original inter-column security game to the "modified" inter-column security game.

Proof of "Modified" Inter-Column Security. We now prove the (computational) security of "modified" inter-column security of our iaiO construction based on subspace hiding assumption-1. Suppose that there exists some PPT adversary $\mathcal{A}$ which achieves non-negligible winning advantage in the "modified" inter-column security game (i,e., where the challenger in the inter-column security game sets the Kilian randomizer matrices corresponding to the challenge programs to be uniformly random but *identical*) 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 $\beta \in \{0,1\}$. We will create a PPT algorithm $\mathcal{B}$ that achieves a non-negligible advantage in breaking a subspace hiding assumption-1 instance parameterized by N, L, T and $Z = 2L$ (where $\mathcal{B}$ uses the same challenge-index i^* and the same challenge permutation-slots (t_0^*, t_1^*) as chosen by $\mathcal{A}$).

Setup. Let $P_1, \ldots, P_T$ be the oblivious branching programs of depth L that the adversary in the inter-column security game chooses, where for each $t \in [T]$, we have

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

subject to the restriction that $P_{t_0^*}$ and $P_{t_1^*}$ are identical. For each $t \in [T]$, the algorithm $\mathcal{B}$ uniformly randomly samples *additional permutation matrices* $Z_{t,1} \ldots, Z_{t,L-1} \in \{0,1\}^{5\times 5}$, where $Z_{t,0}$ and $Z_{t,L}$ are both set to be the identity permutation, subject to the restriction that for each $\ell \in [L]$, we have

$$Z_{t_0^*,\ell} = Z_{t_1^*,\ell}.$$

$\mathcal{B}$ defines a new set of matrices $\{N_{t,\ell,b}\}_{t\in[T],\ell\in[L],b\in\{0,1\}}$, where

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

and sends the following to the challenger in the subspace-hiding assumption-1 game:

$$\left(t_0^*, t_1^*, i^*, \{N_{t,\ell,b}\}_{t\in[T],\ell\in[L],b\in\{0,1\}}\right).$$

Inputs to $\mathcal{B}$. As described in Figure 10 (Section 5), $\mathcal{B}$ receives as input several terms of the following kinds:

1. **Generators:**

   - For each $\ell \in [L]$, $b \in \{0,1\}$ and each $t \in [T] \setminus \{t_1^*\}$: many samples of the form $\mathbf{L}_\ell \mathbf{D}^{(t,N_{t,\ell,b})} \mathbf{R}_\ell$, where $\mathbf{D}^{(t,N_{t,\ell,b})}$ is a permutation-embedded block-diagonal matrix sampled from the appropriate distribution.
   - For each $i \in [N]$ and each $t \in [T]\setminus\{t_1^*\}$: many samples of the form $\mathbf{L}_{L+i} \mathbf{D}^{(t,I)} \mathbf{R}_{L+i}$, where $I \in \{0,1\}^{5\times 5}$ denotes the identity permutation, and $\mathbf{D}^{(t,I)}$ is an identity-permutation-embedded block-diagonal matrix sampled from the appropriate distribution.
   - For each $\ell \in [L+N]$ and each $z \in [2L]$: many samples of the form $\mathbf{L}_\ell \mathbf{D}^{(T+z)} \mathbf{R}_\ell$, where $\mathbf{D}^{(T+z)}$ is a block-diagonal matrix sampled from the appropriate distribution.

   In each case, the "end guards" $\mathbf{L}_1$ and $\mathbf{R}_{L+N}$ are specially structured, as described in Section 4.3.

2. **Challenge-Relevant Elements**:

   - For each $\ell \in [L]$ and for each $b \in \{0,1\}$: many samples of the form

     $$\mathbf{L}_\ell \mathbf{D}^{(t_0^*,N_{t_0^*,\ell,b}),(t_1^*,N_{t_1^*,\ell,b})} \mathbf{R}_\ell,$$

     where $\mathbf{D}^{(t_0^*,N_{t_0^*,\ell,b}),(t_1^*,N_{t_1^*,\ell,b})}$ is a permutation-embedded block-diagonal matrix sampled from the appropriate distribution.
   - For each $i \in [N]$: many samples of the form $\mathbf{L}_{L+i} \mathbf{D}^{(t_0^*,I),(t_1^*,I)} \mathbf{R}_{L+i}$, where $I \in \{0,1\}^{5\times 5}$ denotes the identity permutation, and $\mathbf{D}^{(t_0^*,I),(t_1^*,I)}$ is an identity-permutation-embedded block-diagonal matrix sampled from the appropriate distribution.

   Once again, in each case, the "end guards" $\mathbf{L}_1$ and $\mathbf{R}_{L+N}$ are specially structured, as described in Section 4.3.

3. **Special Challenge-Index Elements**: Many samples of the form

   $$\mathbf{L}_{(L+i^*)} \mathbf{D}^{(t_1^*,I)} \mathbf{R}_{(L+i^*)},$$

   where $I \in \{0,1\}^{5\times 5}$ again denotes the identity permutation, and $\mathbf{D}^{(t_1^*,I)}$ is an identity-permutation-embedded block-diagonal matrix sampled from the appropriate distribution.

   Also, in the case when $i^* = N$, the right "end guard" $\mathbf{R}_{L+N}$ is specially structured, as described in Section 4.3.

- To simulate guarded versions of the program carrying matrices, $\mathcal{B}$ uses the following strategy at each level $\ell \in [L]$ for simulating the block sub-matrices corresponding to the aggregate matrix:
  1. For simulating the block sub-matrices corresponding to any program index $t \in [N] \setminus \{t_0^*, t_1^*\}$, $\mathcal{B}$ appropriately R-linear sums the generator elements of the form $\mathbf{L}_\ell \mathbf{D}^{(t,N_{t,\ell,b})} \mathbf{R}_\ell$ that it receives as input.
  2. For simulating the block sub-matrices corresponding to the program indices t_0^*, t_1^*, $\mathcal{B}$ appropriately R-linear sums the challenge-relevant elements of the form $\mathbf{L}_\ell \mathbf{D}^{(t_0^*,N_{t_0^*,\ell,b}),(t_1^*,N_{t_1^*,\ell,b})} \mathbf{R}_\ell$ that it receives as input.
- To simulate guarded versions of the enforcer matrices, $\mathcal{B}$ appropriately R-linear sums over the generator elements received as input, except for the block sub-matrices corresponding to the diagonal slot t_1^*. $\mathcal{B}$'s simulation strategy for the "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) are 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 $R^{5m \times 5m}$.
  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 block-diagonal matrix in $R^{5m \times 5m}$ and $\mathbf{T}_{t_1^*,i,b}$ as the zero matrix in $R^{5m \times 5m}$. To do this, it additionally R-linear sums over generator matrices of the form $\mathbf{L}_{L+i} \mathbf{D}^{(t_0^*,I)} \mathbf{R}_{L+i}$, 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 block-diagonal matrices in $R^{5m \times 5m}$. To do this, it additionally R-linear sums over the challenge-relevant matrices of the form $\mathbf{L}_{L+i} \mathbf{D}^{(t_0^*,I),(t_1^*,I)} \mathbf{R}_{L+i}$, which it is provided with as input.
  4. **Case-4:** $(i, b) = (i^*, 1-\beta)$, $\mathbf{G}[i, t_0^*, b] = \mathbf{G}[i, t_1^*, b] = 0$: 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 $R^{5m \times 5m}$.
  5. **Case-5:** $(i, b) = (i^*, 1-\beta)$, $\mathbf{G}[i, t_0^*, b] = 1$, $\mathbf{G}[i, t_1^*, b] = 0$: In this case, $\mathcal{B}$ simulates $\mathbf{T}_{t_0^*,i,b}$ as a uniformly distributed block-diagonal matrix in $R^{5m \times 5m}$ and $\mathbf{T}_{t_1^*,i,b}$ as the zero matrix in $R^{5m \times 5m}$. To do this, it additionally R-linear sums over generator matrices of the form $\mathbf{L}_{L+i^*} \mathbf{D}^{(t_0^*,I)} \mathbf{R}_{L+i^*}$, which it is provided with as input.
  6. **Case-6:** $(i, b) = (i^*, 1-\beta)$, $\mathbf{G}[i, t_0^*, b] = 0$, $\mathbf{G}[i, t_1^*, b] = 1$: In this case, $\mathcal{B}$ simulates $\mathbf{T}_{t_0^*,i,b}$ as the zero matrix in $R^{5m \times 5m}$ and $\mathbf{T}_{t_1^*,i,b}$ as a uniformly distributed block-diagonal matrix in $R^{5m \times 5m}$. To do this, it additionally R-linear sums over the special challenge-index matrices of the form $\mathbf{L}_{L+i^*} \mathbf{D}^{(t_1^*,I)} \mathbf{R}_{L+i^*}$, which it is provided with as input.
  7. **Case-7:** $(i, b) = (i^*, 1-\beta)$, $\mathbf{G}[i, t_0^*, b] = \mathbf{G}[i, t_1^*, b] = 1$: In this case, $\mathcal{B}$ simulates both $\mathbf{T}_{t_0^*,i,b}$ and $\mathbf{T}_{t_1^*,i,b}$ as uniformly distributed bloc-diagonal matrices in $R^{5m \times 5m}$. To do this, it additionally R-linear sums over generator matrices of the form $\mathbf{L}_{L+i^*} \mathbf{D}^{(t_0^*,I)} \mathbf{R}_{L+i^*}$ as well as the special challenge-index matrices of the form $\mathbf{L}_{L+i^*} \mathbf{D}^{(t_1^*,I)} \mathbf{R}_{L+i^*}$, all of which it is provided with as input.
  8. **Case-8:** $(i, b) = (i^*, \beta)$: In this case, we must have $\mathbf{G}[i^*, t_0^*, \beta] = 1$ (this follows from the definition of the inter-column security game). In this case, $\mathcal{B}$ uses the challenge term provided to it as input.

Figure 13: The Simulation Strategy used by the Reduction Algorithm $\mathcal{B}$ in the Proof of Modified Inter-Column Security

4. **The Challenge Element**: A sample that has one of the two following forms (b being the choice bit of the challenger in the subspace hiding experiment):
   - $\mathbf{L}_{(L+i^*)} \mathbf{D}^{(t_0^*,I)} \mathbf{R}_{(L+i^*)}$ if $b = 0$, or
   - $\mathbf{L}_{(L+i^*)} \mathbf{D}^{(t_0^*,I),(t_1^*,I)} \mathbf{R}_{(L+i^*)}$ if $b = 1$,

   where $I \in \{0,1\}^{5\times 5}$ again denotes the identity permutation, and $\mathbf{D}^{(t_0^*,I)}$ and $\mathbf{D}^{(t_0^*,I),(t_1^*,I)}$ are block-diagonal matrices sampled from the appropriate distribution.

   Also, in the case when $i^* = N$, the right "end guard" $\mathbf{R}_{L+N}$ is specially structured, as described in Section 4.3.

We structure the rest of the proof as follows. For clarity of exposition, we first state what the challenger in the "modified" inter-column security game should generate, followed by how the algorithm $\mathcal{B}$ simulates the same in a manner that is either computationally or statistically indistinguishable from the real game. The simulation strategy used by $\mathcal{B}$ is as summarized in Figure 13.

Real World: Program-Carrying Matrices. Let the 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:

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

For each level $\ell \in [L]$, and each bit $b \in \{0,1\}$, the challenger in the "modified" inter-column security game generates an "aggregate" block-diagonal matrix $\mathbf{A}_{\ell,b}$, structured as:

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

Based on the above, for each level $\ell \in [L]$ and for each bit $b \in \{0,1\}$, the challenger in the "modified" inter-column security game generates a block diagonal "program-carrying matrix" $\mathbf{P}_{\ell,b}$, of dimension $5(T+2L)m \times 5(T+2L)m$, structured as:

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

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

$$\mathbf{C}_{(\ell,\ell'),(b,b')} = \begin{cases} \mathbf{0} & \text{if } (\ell', b') = (\ell, 1-b), \\ \text{uniform in } R^{5m\times 5m} & \text{otherwise.} \end{cases}.$$

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 sub-matrices, $\mathcal{B}$ simply uses the generator elements that is receives as part if its input. When generating a "guarded" block sub-matrix in a given diagonal slot, it R-linear sums sufficiently many instances of the corresponding generator matrices, which results in the desired distribution by the leftover hash lemma. A zero "guarded" block sub-matrix in a given diagonal slot, on the other hand, conceptually corresponds to R-linear summing the appropriate generator elements with an "all-zero" bit string, or equivalently, not summing any of these elements.

For simulating the program-carrying block sub-matrices within the "aggregate matrix", the natural strategy for $\mathcal{B}$ to adopt is to appropriately R-linear sum the generator elements of the form $\mathbf{L}_\ell \mathbf{D}^{(t,N_{t,\ell,b})}\mathbf{R}_\ell$ 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 $\mathbf{L}_\ell \mathbf{D}^{(t_1^*,N_{t_1^*,\ell,b})}\mathbf{R}_\ell$ 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 Kilian-style randomized permutation matrices at each level (recall that in the modified inter-column security game, the challenger identical Kilian randomizers for both challenge program slots). 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 $\mathbf{L}_\ell \mathbf{D}^{(t_0^*,N_{t_0^*,\ell,b}),(t_1^*,N_{t_1^*,\ell,b})}\mathbf{R}_\ell$.

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

1. For simulating the block sub-matrices corresponding to any program index $t \in [N] \setminus \{t_0^*, t_1^*\}$, $\mathcal{B}$ appropriately R-linear sums the generator elements of the form

$$\mathbf{L}_\ell \mathbf{D}^{(t,N_{t,\ell,b})}\mathbf{R}_\ell,$$

that it receives as input.

2. For simulating the block sub-matrices corresponding to the program indices t_0^*, t_1^*, $\mathcal{B}$ appropriately R-linear sums the challenge-relevant elements of the form

$$\mathbf{L}_\ell \mathbf{D}^{(t_0^*, N_{t_0^*,\ell,b}),(t_1^*, N_{t_1^*,\ell,b})} \mathbf{R}_\ell,$$

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 "modified" 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 "modified" inter-column security game generates an additional "enforcer matrix" $\mathbf{E}_{i,b}$ of dimension $(T+2L)m \times (T+2L)m$, structured as:

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

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

$$\mathbf{C}_{(i,\ell'),(b,b')} = \begin{cases} \mathbf{0} & \text{if } (i,b') = (\phi(\ell), b), \\ \text{uniform in } R^{5m \times 5m} & \text{otherwise.} \end{cases},$$

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 ℓth level branch. Also, the "activation-enforcer" matrix $\mathbf{T}_{i,b}$ is a block diagonal matrix in $R^{5Tm \times 5Tm}$, 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 $\mathbf{T}_{t,i,b}$ satisfies the following distribution:

$$\mathbf{T}_{t,i,b} = \begin{cases} \mathbf{0} & \text{if } \mathbf{G}[i,t,b] = 0, \\ \text{uniform block-diagonal in } R^{5m \times 5m} & \text{otherwise.} \end{cases}.$$

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 $R^{5Tm \times 5Tm}$, 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:

$$\mathbf{T}_{t,i,b} = \begin{cases} \mathbf{0} & \text{if } \mathbf{G}[i,t,b] = 0, \\ \text{block-diagonal uniform in } R^{5m \times 5m} & \text{otherwise.} \end{cases}.$$

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

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 $R^{5m\times 5m}$.

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 block-diagonal matrix in $R^{5m\times 5m}$ and $\mathbf{T}_{t_1^*,i,b}$ as the zero matrix in $R^{5m\times 5m}$. To do this, it additionally R-linear sums over generator matrices of the form $\mathbf{L}_{L+i}\mathbf{D}^{(t_0^*,I)}\mathbf{R}_{L+i}$, 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 block-diagonal matrices in $R^{5m\times 5m}$. To do this, it additionally R-linear sums over the challenge-relevant matrices of the form $\mathbf{L}_{L+i}\mathbf{D}^{(t_0^*,I),(t_1^*,I)}\mathbf{R}_{L+i}$, which it is provided with as input.

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

   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 $R^{5m\times 5m}$.

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

   In this case, $\mathcal{B}$ simulates $\mathbf{T}_{t_0^*,i,b}$ as a uniformly distributed block-diagonal matrix in $R^{5m\times 5m}$ and $\mathbf{T}_{t_1^*,i,b}$ as the zero matrix in $R^{5m\times 5m}$. To do this, it additionally R-linear sums over generator matrices of the form $\mathbf{L}_{L+i^*}\mathbf{D}^{(t_0^*,I)}\mathbf{R}_{L+i^*}$, which it is provided with as input.

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

   In this case, $\mathcal{B}$ simulates $\mathbf{T}_{t_0^*,i,b}$ as the zero matrix in $R^{5m\times 5m}$ and $\mathbf{T}_{t_1^*,i,b}$ as a uniformly distributed block-diagonal matrix in $R^{5m\times 5m}$. To do this, it additionally R-linear sums over the special challenge-index matrices of the form $\mathbf{L}_{L+i^*}\mathbf{D}^{(t_1^*,I)}\mathbf{R}_{L+i^*}$, which it is provided with as input.

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

   In this case, $\mathcal{B}$ simulates both $\mathbf{T}_{t_0^*,i,b}$ and $\mathbf{T}_{t_1^*,i,b}$ as uniformly distributed bloc-diagonal matrices in $R^{5m\times 5m}$. To do this, it additionally R-linear sums over generator matrices of the form $\mathbf{L}_{L+i^*}\mathbf{D}^{(t_0^*,I)}\mathbf{R}_{L+i^*}$ as well as the special challenge-index matrices of the form $\mathbf{L}_{L+i^*}\mathbf{D}^{(t_1^*,I)}\mathbf{R}_{L+i^*}$, all of which it is provided with as input.

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

   In this case, we must have $\mathbf{G}[i^*,t_0^*,\beta] = 1$ (this follows from the definition of the inter-column security game). 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 $\mathbf{L}_{(L+i^*)}\mathbf{D}^{(t_0^*,I)}\mathbf{R}_{(L+i^*)}$, then Case-8 would correspond to the case where $\mathbf{G}[i^*, t_1^*, \beta] = 0$.
2. If the challenge sample is of the form $\mathbf{L}_{(L+i^*)}\mathbf{D}^{(t_0^*,I),(t_1^*,I)}\mathbf{R}_{(L+i^*)}$, then Case-8 would correspond to the case where $\mathbf{G}[i^*, t_1^*, \beta] = 1$. □

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

6.2 Single-Input Program Switching Security

Lemma 6.2. *Assuming that the subspace hiding assumption-2 as described in Definition 5.2 holds, our iaiO construction achieves single-input program switching security as described in Definition 3.7.*

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.

6.2.1 The Hybrid Experiments

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 sub-matrices 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 $R^{5m \times 5m}$.

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.

6.2.2 Proof of Indistinguishability of Hybrids

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 $\beta \in \{0, 1\}$. We use $\mathcal{A}$ to build a PPT algorithm $\mathcal{B}$ that breaks a subspace hiding assumption-2 instance parameterized by N, L, T and $Z = 2L$ (where $\mathcal{B}$ uses the same challenge-index ℓ^*, the challenge-slots $(t^*, z^*) = (t^*, (T + 2\ell^* + \beta))$ and the same challenge bit β) with non-negligible advantage.

Setup. Let $P_1, \ldots, P_T$ be the oblivious branching programs of depth L that the adversary in the inter-column security game chooses, where for each $t \in [T]$, we have

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

For each $t \in [T]$, the algorithm $\mathcal{B}$ uniformly randomly samples *additional permutation matrices* $Z_{t,1} \ldots, Z_{t,L-1} \in \{0,1\}^{5\times 5}$, where $Z_{t,0}$ and $Z_{t,L}$ are both set to be the identity permutation. $\mathcal{B}$ defines a new set of matrices $\{N_{t,\ell,b}\}_{t\in[T],\ell\in[L],b\in\{0,1\}}$, where

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

and sends the following to the challenger in the subspace-hiding assumption-1 game:

$$\left(t^*, (T + 2\ell^* + \beta), \ell^*, \beta, \{N_{t,\ell,b}\}_{t\in[T],\ell\in[L],b\in\{0,1\}}\right).$$

Inputs to $\mathcal{B}$. As described in Figure 11 (Section 5), $\mathcal{B}$ receives as input several terms as follows:

1. **Generators:**

   - For each $\ell \in [L]$, $b \in \{0,1\}$ and each $t \in [T] \setminus \{t^*\}$: many samples of the form $\mathbf{L}_\ell \mathbf{D}^{(t,N_{t,\ell,b})} \mathbf{R}_\ell$, where $\mathbf{D}^{(t,N_{t,\ell,b})}$ is a permutation-embedded block-diagonal matrix sampled from the appropriate distribution.
   - For each $i \in [N]$ and each $t \in [T] \setminus \{t^*\}$: many samples of the form $\mathbf{L}_{L+i} \mathbf{D}^{(t,I)} \mathbf{R}_{L+i}$, where $I \in \{0,1\}^{5\times 5}$ denotes the identity permutation, and $\mathbf{D}^{(t,I)}$ is an identity-permutation-embedded block-diagonal matrix sampled from the appropriate distribution.
   - For each $\ell \in [L+N]$ and each $z \in [Z]$: many samples of the form $\mathbf{L}_\ell \mathbf{D}^{(T+z)} \mathbf{R}_\ell$, where $\mathbf{D}^{(T+z)}$ is a block-diagonal matrix sampled from the appropriate distribution.

   In each case, the "end guards" $\mathbf{L}_1$ and $\mathbf{R}_{L+N}$ are specially structured, as described in Section 4.3.

2. **Challenge-Relevant Elements**:

   - For each $\ell \in [L]$ and $b \in \{0,1\}$: many samples of the form
   $$\mathbf{L}_\ell \mathbf{D}^{(t^*,N_{t^*,\ell,b}),(T+2\ell^*+\beta)} \mathbf{R}_\ell,$$
   where $\mathbf{D}^{(t^*,N_{t^*,\ell,b}),(T+2\ell^*+\beta)}$ is a permutation-embedded block-diagonal matrix sampled from the appropriate distribution, with an R-embedding of the permutation matrix $N_{t^*,\ell,b}$ in the t^*-th diagonal block and a uniform sub-matrix in the $(T+2\ell^*+\beta)$-th diagonal block.
   - For each $i \in [N]$: many samples of the form
   $$\mathbf{L}_{L+i} \mathbf{D}^{(t^*,I),(T+2\ell^*+\beta)} \mathbf{R}_{L+i},$$
   where $I \in \{0,1\}^{5\times 5}$ denotes the identity permutation, and $\mathbf{D}^{(t^*,I),(T+2\ell^*+\beta)}$ is an identity permutation-embedded block-diagonal matrix sampled from the appropriate distribution, with an R-embedding of I in the t^*-th diagonal block and a uniform sub-matrix in the $(T+2\ell^*+\beta)$-th diagonal block.

   Once again, in each case, the "end guards" $\mathbf{L}_1$ and $\mathbf{R}_{L+N}$ are specially structured, as described in Section 4.3.

3. **Special Challenge-Index Elements**: Many samples of the form $\mathbf{L}_{\ell^*} \mathbf{D}^{(t^*,N_{t^*,\ell^*,\beta})} \mathbf{R}_{\ell^*}$, where $\mathbf{D}^{(t^*,P_{N^*,\ell^*,\beta})}$ is a permutation-embedded block-diagonal matrix sampled from the appropriate distribution.

   Also, in the case when $\ell^* = 1$, the left "end guard" $\mathbf{L}_1$ is specially structured, as described in Section 4.3.

4. **The Challenge Element**: A sample that has one of the two following forms (b being the bit sampled internally by the challenger in the subspace hiding experiment):

   - $\mathbf{L}_{\ell^*} \mathbf{D}^{(T+2\ell^*+\beta)} \mathbf{R}_{\ell^*}$ if $b = 0$, or
   - $\mathbf{L}_{\ell^*} \mathbf{D}^{(t^*,N_{t^*,\ell^*,\beta}),(T+2\ell^*+\beta)} \mathbf{R}_{\ell^*}$ if $b = 1$,

   where:

   - $\mathbf{D}^{(T+2\ell^*+\beta)}$ is a block-diagonal matrix sampled from the appropriate distribution, with a uniform sub-matrix in the $(T+2\ell^*+\beta)$-th diagonal block.
   - $\mathbf{D}^{(t^*,N_{t^*,\ell,\beta}),(T+2\ell^*+\beta)}$ is a permutation-embedded block-diagonal matrix sampled from the appropriate distribution, with an R-embedding of the permutation matrix $N_{t^*,\ell^*,\beta}$ in the t^*-th diagonal block and a uniform sub-matrix in the $(T+2\ell^*+\beta)$-th diagonal block.

   Also, in the case when $\ell^* = 1$, the left "end guard" $\mathbf{L}_1$ is specially structured, as described in Section 4.3.

We structure the rest of the proof as follows. For clarity of exposition, we first state what the challenger in the single-input program switching security game should generate, followed by how the algorithm $\mathcal{B}$ simulates the same in a manner that is either computationally or statistically indistinguishable from the real game. The simulation strategy used by $\mathcal{B}$ is as summarized in Figure 14.

- $\mathcal{B}$ simulates guarded versions of the program-carrying matrices by appropriately R-linear summing over the generator, except for the block sub-matrix corresponding to the diagonal slot t^*. $\mathcal{B}$'s strategy to simulate the block matrix $\mathbf{N}_{t^*,\ell,b}$ is as described below:
  1. **Case-1:** $\ell \neq \ell^*$ **and** $b = x^*_{\phi(\ell)}$: In this case, $\mathcal{B}$ appropriately R-linear sums over the following input elements: (a) the challenge-relevant elements $\mathbf{L}_\ell \mathbf{D}^{(t^*, N_{t^*,\ell,b}),(T+2\ell^*+\beta)} \mathbf{R}_\ell$, and (b) all other generator elements, *including* the generator elements of the form $\mathbf{L}_\ell \mathbf{D}^{(T+2\ell^*+\beta)} \mathbf{R}_\ell$.
  2. **Case-2:** $\ell > \ell^*$ **and** $b = 1 - x^*_{\phi(\ell)}$: In this case, $\mathcal{B}$ again appropriately R-linear sums over the following input elements: (a) the challenge-relevant elements $\mathbf{L}_\ell \mathbf{D}^{(t^*, N_{t^*,\ell,b}),(T+2\ell^*+\beta)} \mathbf{R}_\ell$, and (b) all other generator elements, *including* the generator elements of the form $\mathbf{L}_\ell \mathbf{D}^{(T+2\ell^*+\beta)} \mathbf{R}_\ell$.
  3. **Case-3:** $\ell < \ell^*$ **and** $b = 1 - x^*_{\phi(\ell)}$: In this case, $\mathcal{B}$ appropriately R-linear sums over only the generator elements it received as input, *including* the generator elements of the form $\mathbf{L}_\ell \mathbf{D}^{(T+2\ell^*+\beta)} \mathbf{R}_\ell$.
  4. **Case-4:** $\ell = \ell^*$ **and** $b = 1 - \beta$: In this case, $\mathcal{B}$ appropriately R-linear sums over the following input elements: (a) the "special" challenge-index elements $\mathbf{L}_{\ell^*} \mathbf{D}^{(t^*, N_{t^*,\ell,b})} \mathbf{R}_{\ell^*}$, and (b) all generator elements *excluding* the generator elements of the form $\mathbf{L}_{\ell^*} \mathbf{D}^{(T+2\ell^*+\beta)} \mathbf{R}_{\ell^*}$.
  5. **Case-5:** $\ell = \ell^*$ **and** $b = \beta$: In this case, $\mathcal{B}$ directly uses the challenge term provided to it as input.
- $\mathcal{B}$ simulates guarded versions of the enforcer matrices by appropriately R-linear summing over the generator elements received as input, except for the block sub-matrix corresponding to the diagonal slot t^*. $\mathcal{B}$'s strategy to simulate the enforcer block matrices $\mathbf{T}_{t^*,i,b}$ for different values of (i, b) is as described below:
  1. **Case-1:** $\mathbf{G}[i, t^*, b] = 0$: In this case, $\mathcal{B}$ appropriately R-linear sums over only the generator elements it received as input, *including* the generator element $\mathbf{L}_{L+i} \mathbf{D}^{(T+2\ell^*+\beta)} \mathbf{R}_{L+i}$.
  2. **Case-2:** $\mathbf{G}[i, t^*, b] = 1$: In this case, $\mathcal{B}$ appropriately R-linear sums over the following input elements: (a) the challenge-relevant elements of the form $\mathbf{L}_{L+i} \mathbf{D}^{(t^*, I),(T+2\ell^*+\beta)} \mathbf{R}_{L+i}$, and (b) all generator elements *including* the generator element $\mathbf{L}_{L+i} \mathbf{D}^{(T+2\ell^*+\beta)} \mathbf{R}_{L+i}$.

Figure 14: The Simulation Strategy used by the Reduction Algorithm $\mathcal{B}$ in the Proof of Indistinguishability of Hybrids for Single-Input Program Switching Security Proof

Real World: Program-Carrying Matrices. Let the 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:

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

For each level $\ell \in [L]$, and each bit $b \in \{0, 1\}$, the challenger in the "modified" inter-column security game generates an "aggregate" block-diagonal matrix $\mathbf{A}_{\ell,b}$, structured as:

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

Based on the above, for each level $\ell \in [L]$ and for each bit $b \in \{0, 1\}$, the challenger in the single-input program switching security game generates a block diagonal "program-carrying matrix" $\mathbf{P}_{\ell,b}$, of dimension $5(T + 2L)m \times$

$5(T+2L)m$, structured as:

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

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

$$\mathbf{C}_{(\ell,\ell'),(b,b')} = \begin{cases} \mathbf{0} & \text{if } (\ell', b') = (\ell, 1-b), \\ \text{uniform in } R^{5m \times 5m} & \text{otherwise.} \end{cases}.$$

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 $\mathbf{A}_{\ell,b}$, structured as:

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

except for the block sub-matrix corresponding to the diagonal slot t^*. In particular, by appropriately R-linear summing over the generator elements received as input, $\mathcal{B}$ can simulate "guarded" versions of all block sub-matrices in the "aggregate" block-diagonal matrix, except for the block sub-matrix corresponding to the diagonal slot t^*. The reason $\mathcal{B}$ cannot simulate this sub-matrix is that by definition of the subspace hiding assumption, it is not provided with the generator elements corresponding to the diagonal slot t^*.

The crux of the reduction lies in how $\mathcal{B}$ simulates the block matrix $\mathbf{N}_{t^*,\ell,b}$ 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 ℓth level branch):

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

   In this case, $\mathcal{B}$ should set $\mathbf{N}_{t^*,\ell,b}$ to be an R-embedding of $N_{t^*,\ell,b}$, and the consistency check matrix $\mathbf{C}_{\ell,1-b}$ to the zero matrix in $R^{5m \times 5m}$. Also, the consistency check matrix $\mathbf{C}_{\ell^*,\beta}$ is set to uniformly random. Hence, to simulate this, $\mathcal{B}$ can appropriately R-linear sum over the following input elements: (a) the challenge-relevant elements $\mathbf{L}_\ell \mathbf{D}^{(t^*, N_{t^*,\ell,b}),(T+2\ell^*+\beta)} \mathbf{R}_\ell$, and (b) all other generator elements, *including* the generator elements of the form $\mathbf{L}_\ell \mathbf{D}^{(T+2\ell^*+\beta)} \mathbf{R}_\ell$.

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

   Note that even in this case, $\mathcal{B}$ should set $\mathbf{N}_{t^*,\ell,b}$ to be an R-embedding of $N_{t^*,\ell,b}$ and the consistency check matrix $\mathbf{C}_{\ell,1-b}$ to the zero matrix in $R^{5m \times 5m}$. Also, once again, the consistency check matrix $\mathbf{C}_{\ell^*,\beta}$ is set to uniformly random. Hence, to simulate this, $\mathcal{B}$ can again appropriately R-linear sum over the following input elements: (a) the challenge-relevant elements $\mathbf{L}_\ell \mathbf{D}^{(t^*, N_{t^*,\ell,b}),(T+2\ell^*+\beta)} \mathbf{R}_\ell$, and (b) all other generator elements, *including* the generator elements of the form $\mathbf{L}_\ell \mathbf{D}^{(T+2\ell^*+\beta)} \mathbf{R}_\ell$.

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

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

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

   In this case, $\mathcal{B}$ should set $\mathbf{N}_{t^*,\ell,b}$ to be an R-embedding of $N_{t^*,\ell,b}$, and the consistency check matrix $\mathbf{C}_{\ell^*,\beta}$ to the zero matrix in $R^{5m\times 5m}$. Hence, to simulate this, $\mathcal{B}$ can appropriately R-linear sum over the following input elements: (a) the "special" challenge-index elements $\mathbf{L}_{\ell^*}\mathbf{D}^{(t^*,N_{t^*,\ell,b})}\mathbf{R}_{\ell^*}$, and (b) all generator elements *excluding* the generator elements of the form $\mathbf{L}_{\ell^*}\mathbf{D}^{(T+2\ell^*+\beta)}\mathbf{R}_{\ell^*}$.

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

   In this case, $\mathcal{B}$ uses the challenge term provided to it as input. Recall that this is a sample that has one of the two following forms:

   - $\mathbf{L}_{\ell^*}\mathbf{D}^{(T+2\ell^*+\beta)}\mathbf{R}_{\ell^*}$, or
   - $\mathbf{L}_{\ell^*}\mathbf{D}^{(t^*,N_{t^*,\ell^*,\beta}),(T+2\ell^*+\beta)}\mathbf{R}_{\ell^*}$,

   Note that this is an acceptable simulation strategy since no matter what the form of the challenge term is, $\mathcal{B}$ implicitly sets the consistency check matrix $\mathbf{C}_{\ell^*,\beta}$ to uniformly random, as desired in this case.

At this point, observe the following:

1. If the challenge sample is of the form $\mathbf{L}_{\ell^*}\mathbf{D}^{(T+2\ell^*+\beta)}\mathbf{R}_{\ell^*}$, then so far, $\mathcal{B}$ has properly simulated "guarded" versions of the program-carrying matrices as per Expt_{ℓ^*}; in particular, it has "erased" the matrix $N_{t^*,\ell^*,\beta}$ by implicitly setting its ring-embedded version to 0.
2. If the challenge sample is of the form $\mathbf{L}_{\ell^*}\mathbf{D}^{(t^*,N_{t^*,\ell^*,\beta}),(T+2\ell^*+\beta)}\mathbf{R}_{\ell^*}$, then so far, $\mathcal{B}$ has properly simulated "guarded" versions of the program-carrying matrices as per Expt_{ℓ^*-1}; in particular, it has *not* "erased" the matrix $N_{t^*,\ell^*,\beta}$.

Next we describe how $\mathcal{B}$ simulates the enforcer matrices. Once again, for clarity of exposition, we first state what the challenger in the single-input program switching security game should generate, followed by how the algorithm $\mathcal{B}$ simulates the same in a manner that is either computationally or statistically indistinguishable from the real 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 program switching security game generates an additional "enforcer matrix" $\mathbf{E}_{i,b}$ of dimension $(T+2L)m \times (T+2L)m$, structured as:

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

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 ℓth level branch. Also, the "activation-enforcer" matrix $\mathbf{T}_{i,b}$ is a block diagonal matrix in $R^{5Tm\times 5Tm}$, 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 $\mathbf{T}_{t,i,b}$ satisfies the following distribution:

$$\mathbf{T}_{t,i,b} = \begin{cases} \mathbf{0} & \text{if } \mathbf{G}[i,t,b] = 0, \\ \text{uniform block-diagonal in } R^{5m\times 5m} & \text{otherwise.} \end{cases}.$$

Simulation: Enforcer Matrices. In this simulation, $\mathcal{B}$ (implicitly) structures the "activation-enforcer" matrix $\mathbf{T}_{i,b}$ as a block diagonal matrix in $R^{5Tm \times 5Tm}$, 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:

$$\mathbf{T}_{t,i,b} = \begin{cases} \mathbf{0} & \text{if } \mathbf{G}[i,t,b] = 0, \\ \text{uniform block diagonal in } R^{5m \times 5m} & \text{otherwise.} \end{cases},$$

In particular, by appropriately R-linear summing over the generator elements received as input, $\mathcal{B}$ can simulate "guarded" versions of all consistency sub-matrices and all block sub-matrices in the "activation-enforcer" matrix $\mathbf{T}_{i,b}$, except for the block sub-matrix corresponding to the diagonal slot t^*. The reason $\mathcal{B}$ cannot simulate this sub-matrix is that by definition of the subspace hiding assumption-2, it is not provided with the generator elements corresponding to the diagonal slot t^*.

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

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

   In this case, $\mathcal{B}$ should set $\mathbf{T}_{t^*,i,b}$ to the zero matrix in $R^{5m \times 5m}$. Hence, to simulate this, it suffices for $\mathcal{B}$ to appropriately R-linear sum over only the generator elements it received as input, *including* the generator element $\mathbf{L}_{L+i}\mathbf{D}^{(T+2\ell^*+\beta)}\mathbf{R}_{L+i}$.

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

   In this case, it must be that either $\phi(\ell^*) \neq i$ or $\beta \neq b$. 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 $\mathbf{G}[\phi(\ell^*), t^*, 1-\beta] = 1$ by definition of the single-input switching security game. Hence, in these cases, $\mathcal{B}$ should simulate $\mathbf{T}_{t^*,i,b}$ as a uniformly distributed block-diagonal matrix in $R^{5m \times 5m}$.

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

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 $\mathbf{C}_{\ell^*,\beta}$ is set to a uniformly distributed matrix in $R^{5m \times 5m}$, as either $\phi(\ell^*) \neq i$ or $\beta \neq b$. Recall that in our parallel iO construction, for any enforcer matrix $\mathbf{E}_{i,b}$, a consistency check sub-matrix $\mathbf{C}_{\ell,b'}$ should be set to a uniformly distributed matrix in $R^{5m \times 5m}$ 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 $\mathbf{L}_{\ell^*}\mathbf{D}^{(T+2\ell^*+\beta)}\mathbf{R}_{\ell^*}$ 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 $\mathbf{L}_{\ell^*}\mathbf{D}^{(t^*,N_{t^*,\ell^*,\beta}),(T+2\ell^*+\beta)}\mathbf{R}_{\ell^*}$, 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 immediately.

6.2.3 Finishing the Proof

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 6.3. *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 $Z_1, \ldots, Z_{L-1} \in \{0,1\}^{5\times5}$ 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 $\mathbf{x}$.

Remark 6.4. Note that we can apply this lemma since the Kilian randomizer matrices are sampled independently for each of the T programs.

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. □

6.3 Completely Inactive Program Security and Intra-Column Security

Lemma 6.5. *Assuming that the subspace hiding assumption-2 as described in Definition 5.2 holds, our iaiO construction achieves completely inactive program security as described in Definition 3.7.*

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 based on Kilian randomization, 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 6.6. *Assuming that the subspace hiding assumption-1 as described in Definition 5.1 holds, our iaiO construction achieves intra-column security as described in Definition 3.7.*

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.

7 Proof Of Theorem 5.6

In this section, we prove Theorem 5.6. The proof is sub-divided into several subsections that are organized as follows:

- In Section 7.2, we show that the baby subspace hiding assumption-1 as in Definition 5.3 (resp., the baby subspace hiding assumption-2 as in Definition 5.4) over any ring R implies the subspace hiding assumption-1 as in Definition 5.1 (resp., the subspace hiding assumption-2 as in Definition 5.2) over the same ring R. This is the first step in proving Theorem 5.6.
- In Sections 7.4 and 7.5, we prove that the baby subspace hiding assumptions-1 and 2 as in Definitions 5.3 and 5.4, respectively, hold over any generic ring R such that for any $m, Q = \mathrm{poly}(\lambda)$ where m is any arbitrary positive integer satisfying $m > 3\lambda$ and Q is any arbitrary positive integer, we have

$$(\mathbf{A}, \mathbf{AK}) \overset{c}{\approx} (\mathbf{A}, \mathbf{U}), \quad (\mathbf{B}, \mathbf{K}'\mathbf{B}) \overset{c}{\approx} (\mathbf{B}, \mathbf{V})$$

  where $\mathbf{K}, \mathbf{K}' \leftarrow R^{m\times m}$, $\mathbf{A}, \mathbf{U} \leftarrow R^{Qm\times m}$, and $\mathbf{B}, \mathbf{V} \leftarrow R^{m\times Qm}$. This completes the proof of Theorem 5.6.

Along the way, we introduce several technical tools that facilitate our proofs:

- Our proofs for "lifting" the baby subspace hiding assumptions to their corresponding full versions in Section 7.2 rely on a generic lemma called the "lifting lemma", which we introduce and prove in Section 7.1. The lifting lemma is also crucially used in proving the baby subspace hiding assumptions in Sections 7.4 and 7.5.
- Our proofs of the baby subspace hiding assumptions in Sections 7.4 and 7.5 also rely on proving a simplified version of these assumptions first, which we refer to as the "auxiliary baby subspace hiding assumption. This assumption is defined and proved formally in Section 7.3.

7.1 Lifting Lemma

In this section, we state and prove a general lemma over a ring R (such that R is the output space of an RHS) that we call the "lifting lemma". We subsequently use this lemma to prove several theorems.

7.1.1 Some Distributions

We define the following distributions over the ring R for any parameters $m, m', n, \ell, L, Z \in \mathbb{N}$ such that $L > \ell$, $m \geq 6nL\log|R|$, and $m' > 6m\log|R|$:

- For each $i \in [Z]$ and each $j \in [\ell]$, let $\mathbf{R}_{i,j} \in R^{m\times n}$ and $\mathbf{L}_{i,j} \in R^{n\times m}$ be uniformly random matrices subject to the restriction that for each $i \in [Z-1]$ and for each $j, j' \in [\ell]$ such that $j \neq j'$, we have

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

  for some *block-diagonal* matrix $\mathbf{D}_{i,j} \in R^{m\times m}$.
- For each $i \in [Z]$, let $\mathbf{R}_i \in R^{m\times n\ell}$ and $\mathbf{L}_i \in R^{n\ell\times m}$ be matrices defined as:

$$\mathbf{R}_i = \begin{bmatrix}\mathbf{R}_{i,1} & \dots & \mathbf{R}_{i,\ell}\end{bmatrix}, \quad \mathbf{L}_i = \begin{bmatrix}\mathbf{L}_{i,1}\\ \vdots \\ \mathbf{L}_{i,\ell}\end{bmatrix}.$$

  Observe that for each $i \in [Z-1]$, we have $\mathbf{R}_i\mathbf{L}_{i+1} = \mathbf{D}_i$ for some *block-diagonal* matrix $\mathbf{D}_i \in R^{m\times m}$.

- For any $i \in [Z]$ and any $\mathbf{B} \in R^{n\ell \times n\ell}$ such that $\mathbf{B}$ is n-"block-wise random", i.e., $\mathbf{B}$ is of the form

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

where each of the $\ell \times \ell$ sub-matrices is of size $n \times n$, and is either zero or uniformly random in $R^{n\times n}$, let $\mathbf{M}_{i,\mathbf{B}}$ be a matrix defined as

$$\mathbf{M}_{i,\mathbf{B}} = \mathbf{L}_i \mathbf{B} \mathbf{R}_i.$$

- For each $i \in [Z]$ and each $j \in [L]$, let $\widetilde{\mathbf{R}}_{i,j} \in R^{m' \times n}$ and $\widetilde{\mathbf{L}}_{i,j} \in R^{n \times m'}$ be uniformly random matrices subject to the restriction that for each $i \in [Z-1]$ and for each $j, j' \in [\ell]$ such that $j \neq j'$, we have

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

for some *block-diagonal* matrix $\mathbf{D}_{i,j} \in R^{m' \times m'}$.

- For each $i \in [Z]$, let $\widetilde{\mathbf{R}}_i \in R^{m \times nL}$ and $\widetilde{\mathbf{L}}_i \in R^{nL \times m}$ be matrices defined as:

$$\widetilde{\mathbf{R}}_i = \begin{bmatrix} \widetilde{\mathbf{R}}_{i,1} & \dots & \widetilde{\mathbf{R}}_{i,\ell} \end{bmatrix}, \quad \widetilde{\mathbf{L}}_i = \begin{bmatrix} \widetilde{\mathbf{L}}_{i,1} \\ \vdots \\ \widetilde{\mathbf{L}}_{i,\ell} \end{bmatrix}.$$

Observe that for each $i \in [Z-1]$, we have $\widetilde{\mathbf{R}}_i \widetilde{\mathbf{L}}_{i+1} = \widetilde{\mathbf{D}}_i$, where $\widetilde{\mathbf{D}}_i$ is again some *block-diagonal* matrix in $R^{m' \times m'}$.

- For any $i \in [Z]$, any matrix $\mathbf{B} \in R^{n\ell \times n\ell}$ that is n-"block-wise random", and any matrix $\widetilde{\mathbf{B}} \in R^{n(L-\ell) \times n(L-\ell)}$ that is similarly n-"block-wise random", let $\mathbf{M}_{i,\mathbf{B},\widetilde{\mathbf{B}}}$ be a matrix defined as

$$\mathbf{M}_{i,\mathbf{B},\widetilde{\mathbf{B}}} = \widetilde{\mathbf{L}}_i \begin{bmatrix} \mathbf{B} & \mathbf{0}_R \\ \mathbf{0}_R & \widetilde{\mathbf{B}} \end{bmatrix} \widetilde{\mathbf{R}}_i.$$

- For any $i \in [Z]$, any $\mathbf{B} \in R^{n\ell \times n\ell}$ that is n-"block-wise random", any $\widetilde{\mathbf{B}} \in R^{n(L-\ell) \times n(L-\ell)}$ that is similarly n-"block-wise random", and any pair of permutations $\pi_0, \pi_1 \in S_L$ (where S_L denotes the symmetric group of L elements), let $\mathbf{M}_{i,\mathbf{B},\widetilde{\mathbf{B}},\pi_0,\pi_1}$ be a matrix defined as

$$\mathbf{M}_{i,\mathbf{B},\widetilde{\mathbf{B}},\pi_L,\pi_R} = \widetilde{\mathbf{L}}_i \bar{\pi}_0 \begin{bmatrix} \mathbf{B} & \mathbf{0}_R \\ \mathbf{0}_R & \widetilde{\mathbf{B}} \end{bmatrix} \bar{\pi}_1 \widetilde{\mathbf{R}}_i,$$

where $\bar{\pi}_0 \in R^{nL \times nL}$ and $\bar{\pi}_1 \in R^{nL \times nL}$ are the n-"block-wise" R-embedded matrix representations of π_0 and π_1, respectively.

7.1.2 A Simple Version of the Lifting Lemma

We first state and prove a simple version of the lifting lemma. We subsequently present a generalized version of this lemma.

Lemma 7.1. (Lifting Lemma (Simple).) *For any $T, T' = \mathrm{poly}(\lambda)$, there exists a PPT algorithm that given:*

- *A set of terms of the form* $\{\mathbf{M}_{i,\mathbf{B}_t}\}_{i\in[Z],t\in[T]}$ *where each* $\mathbf{B}_t$ *is an* n*-"blockwise-random" matrix in* $R^{n\ell\times n\ell}$ *and each* $\mathbf{M}_{i,\mathbf{B}_t}$ *is a matrix of the form*

$$\mathbf{M}_{i,\mathbf{B}_t} = \mathbf{L}_i\mathbf{B}_t\mathbf{R}_i,$$

with $\mathbf{L}_i$ *and* $\mathbf{R}_i$ *for* $i \in [Z]$ *distributed as described above, i.e., we have*

$$\mathbf{R}_i = \begin{bmatrix}\mathbf{R}_{i,1} & \dots & \mathbf{R}_{i,\ell}\end{bmatrix}, \quad \mathbf{L}_i = \begin{bmatrix}\mathbf{L}_{i,1}\\ \vdots \\ \mathbf{L}_{i,\ell}\end{bmatrix},$$

where for each $i \in [Z]$ *and each* $j \in [\ell]$, $\mathbf{R}_{i,j} \in R^{m\times n}$ *and* $\mathbf{L}_{i,j} \in R^{n\times m}$ *are uniformly random matrices subject to the restriction that for each* $i \in [Z-1]$ *and for each* $j, j' \in [\ell]$ *such that* $j \neq j'$*, we have*

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

for some block-diagonal matrix $\mathbf{D}_{i,j} \in R^{m\times m}$.

- *A set of terms of the form* $\{\widetilde{\mathbf{B}}_{t'}\}$ *where each* $\widetilde{\mathbf{B}}_{t'}$ *is an* n*-"blockwise-random" matrix in* $R^{n(L-\ell)\times n(L-\ell)}$,

outputs a set of terms of the form $\{\mathbf{M}_{i,\mathbf{B}_t,\widetilde{\mathbf{B}}_{t'}}\}_{i\in[Z],t\in[T],t'\in[T']}$, *where each* $\mathbf{M}_{i,\mathbf{B}_t,\widetilde{\mathbf{B}}_{t'}}$ *is a matrix of the form*

$$\mathbf{M}_{i,\mathbf{B}_t,\widetilde{\mathbf{B}}_{t'}} = \widetilde{\mathbf{L}}_i \begin{bmatrix}\mathbf{B}_t & \mathbf{0}_R\\ \mathbf{0}_R & \widetilde{\mathbf{B}}_{t'}\end{bmatrix} \widetilde{\mathbf{R}}_i,$$

with $\widetilde{\mathbf{L}}_i$ *and* $\widetilde{\mathbf{R}}_i$ *for* $i \in [Z]$ *also distributed as described above, i.e., we have*

$$\widetilde{\mathbf{R}}_i = \begin{bmatrix}\widetilde{\mathbf{R}}_{i,1} & \dots & \widetilde{\mathbf{R}}_{i,\ell}\end{bmatrix}, \quad \widetilde{\mathbf{L}}_i = \begin{bmatrix}\widetilde{\mathbf{L}}_{i,1}\\ \vdots \\ \widetilde{\mathbf{L}}_{i,\ell}\end{bmatrix},$$

where for each $i \in [Z]$ *and each* $j \in [\ell]$, $\widetilde{\mathbf{R}}_{i,j} \in R^{m'\times n}$ *and* $\widetilde{\mathbf{L}}_{i,j} \in R^{n\times m'}$ *are uniformly random matrices subject to the restriction that for each* $i \in [Z-1]$ *and for each* $j, j' \in [\ell]$ *such that* $j \neq j'$*, we have*

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

for some block-diagonal matrix $\mathbf{D}_{i,j} \in R^{m'\times m'}$.

Proof. We prove this by showing the existence of such an algorithm. The algorithm proceeds through the following steps:

1. For each $i \in [Z]$ and each $j \in [\ell+1, L]$, sample additional matrices $\mathbf{R}_{i,j} \in R^{m\times n}$ and $\mathbf{L}_{i,j} \in R^{n\times m}$ subject to the restriction that for each $i \in [Z-1]$ and for each $j, j' \in [\ell+1, L]$ such that $j \neq j'$, we have

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

   for some uniformly random block-diagonal matrix $\mathbf{D}_{i,j} \in R^{m\times m}$. Note that this can be done efficiently using the same procedure as sampling the guard matrices for our iaiO construction in Section 4.

2. Set the following (hypothetically) for each $i \in [Z]$:

   $$\widehat{\mathbf{R}}_i = \begin{bmatrix}\mathbf{R}_{i,1} & \dots & \mathbf{R}_{i,\ell} & \mathbf{R}_{i,\ell+1} & \dots & \mathbf{R}_{i,L}\end{bmatrix}, \quad \widehat{\mathbf{L}}_i = \begin{bmatrix}\mathbf{L}_{i,1}\\ \vdots \\ \mathbf{L}_{i,\ell}\\ \mathbf{L}_{i,\ell+1}\\ \vdots \\ \mathbf{L}_{i,L}\end{bmatrix}.$$

3. For any $i \in [Z], t \in [T], t' \in [T']$, given $\mathbf{M}_{i,\mathbf{B}_t} = \mathbf{L}_i \mathbf{B}_t \mathbf{R}_i$, simulate

$$\widehat{\mathbf{M}}_{i,\mathbf{B}_t,\widetilde{\mathbf{B}}_{t'}} := \widehat{\mathbf{L}}_i \begin{bmatrix} \mathbf{B}_t & \mathbf{0}_R \\ \mathbf{0}_R & \widetilde{\mathbf{B}}_{t'} \end{bmatrix} \widehat{\mathbf{R}}_i$$

as:

$$\widehat{\mathbf{M}}_{i,\mathbf{B}_t,\widetilde{\mathbf{B}}_{t'}} = \mathbf{M}_{i,\mathbf{B}_t} + \begin{bmatrix} \mathbf{R}_{i,\ell+1} & \dots & \mathbf{R}_{i,L} \end{bmatrix} \widetilde{\mathbf{B}}_{t'} \begin{bmatrix} \mathbf{L}_{i,\ell+1} \\ \vdots \\ \mathbf{L}_{i,L} \end{bmatrix}.$$

4. For each $i \in [Z]$ and each $j \in [\ell+1, L]$, sample uniformly random matrices $\mathbf{R}_i^* \in R^{m' \times m}$ and $\mathbf{L}_i^* \in R^{m \times m'}$ subject to the restriction that for each $i \in [Z-1]$, we have

$$\mathbf{R}_i^* \mathbf{L}_{i+1}^* = \mathbf{D}_i^*,$$

for some uniformly random block-diagonal matrix $\mathbf{D}_i^* \in R^{m' \times m'}$. This is again done in exactly the same way as sampling the guard matrices for our iaiO construction in Section 4.

5. Set the following (hypothetically) for each $i \in [Z]$:

$$\widetilde{\mathbf{R}}_i = \widehat{\mathbf{R}}_i \mathbf{R}_i^*, \quad \widetilde{\mathbf{L}}_i = \mathbf{L}_i^* \widehat{\mathbf{L}}_i,$$

where $\widehat{\mathbf{R}}_i$ and $\widehat{\mathbf{L}}_i$ are as set (hypothetically) in Step-3.

6. For any $i \in [Z], t \in [T], t' \in [T']$, given $\mathbf{M}_{i,\mathbf{B}_t} = \mathbf{L}_i \mathbf{B}_t \mathbf{R}_i$, simulate

$$\widetilde{\mathbf{M}}_{i,\mathbf{B}_t,\widetilde{\mathbf{B}}_{t'}} := \widetilde{\mathbf{L}}_i \begin{bmatrix} \mathbf{B}_t & \mathbf{0}_R \\ \mathbf{0}_R & \widetilde{\mathbf{B}}_{t'} \end{bmatrix} \widetilde{\mathbf{R}}_i$$

as:

$$\widetilde{\mathbf{M}}_{i,\mathbf{B}_t,\widetilde{\mathbf{B}}_{t'}} = \mathbf{L}_i^* \widehat{\mathbf{M}}_{i,\mathbf{B}_t,\widetilde{\mathbf{B}}_{t'}} \mathbf{R}_i^* = \mathbf{L}_i^* \left(\mathbf{M}_{i,\mathbf{B}_t} + \begin{bmatrix} \mathbf{R}_{i,\ell+1} & \dots & \mathbf{R}_{i,L} \end{bmatrix} \widetilde{\mathbf{B}}_{t'} \begin{bmatrix} \mathbf{L}_{i,\ell+1} \\ \vdots \\ \mathbf{L}_{i,L} \end{bmatrix} \right) \mathbf{R}_i^*.$$

First, note that for each $i \in [Z-1]$, we have $\widetilde{\mathbf{R}}_i \widetilde{\mathbf{L}}_{i+1} = \widetilde{\mathbf{D}}_i$ for some block-diagonal $\widetilde{\mathbf{D}}_i \in R^{m' \times m'}$, as desired. In order to complete the proof, all we need to do is show that the simulated "mega-guards" of the form $\widetilde{\mathbf{R}}_i = \widehat{\mathbf{R}}_i \mathbf{R}_i^*$ and $\widetilde{\mathbf{L}}_i = \mathbf{L}_i^* \widehat{\mathbf{L}}_i$ are distributed appropriately. We use Lemma 3.10 and Corollary A.3 to argue this:

- First note that, by Lemma 3.10, we can write that for each $i \in [Z-1]$, we have

$$(\widetilde{\mathbf{R}}_i, \widetilde{\mathbf{L}}_{i+1}) \overset{c}{\approx} \left(\begin{bmatrix} \mathbf{A} & \mathbf{AX} + \mathbf{C} \end{bmatrix}, \begin{bmatrix} -\mathbf{XB} \\ \mathbf{B} \end{bmatrix} \right),$$

for uniformly random $\mathbf{A}$, $\mathbf{X}$, and $\mathbf{B}$.

- By Corollary A.3 it follows that for each $i \in [Z-1]$, the matrices $\widetilde{\mathbf{R}}_i$ and $\widetilde{\mathbf{L}}_{i+1}$ are distributed indistinguishably from uniformly sampled random matrices with the desired product distribution, and the proof is complete.

This completes the proof of Lemma 7.1. □

7.1.3 A Generalized Version of the Lifting Lemma

We now state a generalized version of the aforementioned lemma that additionally allows the internal matrices of the form

$$\begin{bmatrix} \mathbf{B}_t & \mathbf{0}_R \\ \\ \mathbf{0}_R & \widetilde{\mathbf{B}}_{t'} \end{bmatrix},$$

in the output distribution to be permuted along the rows and/or columns.

Lemma 7.2. (Lifting Lemma (General).) *For any $T, T' = \mathrm{poly}(\lambda)$, there exists a PPT algorithm that given:*

- *A set of terms of the form $\{\mathbf{M}_{i,\mathbf{B}_t}\}_{i\in[Z],t\in[T]}$ where each $\mathbf{B}_t$ is an n-"blockwise-random" matrix in $R^{n\ell\times n\ell}$ and each $\mathbf{M}_{i,\mathbf{B}_t}$ is a matrix of the form*

$$\mathbf{M}_{i,\mathbf{B}_t} = \mathbf{L}_i\mathbf{B}_t\mathbf{R}_i,$$

  with $\mathbf{L}_i$ and $\mathbf{R}_i$ for $i \in [Z]$ distributed as described above, i.e., we have

$$\mathbf{R}_i = \begin{bmatrix}\mathbf{R}_{i,1} & \dots & \mathbf{R}_{i,\ell}\end{bmatrix}, \quad \mathbf{L}_i = \begin{bmatrix}\mathbf{L}_{i,1} \\ \vdots \\ \mathbf{L}_{i,\ell}\end{bmatrix},$$

  where for each $i \in [Z]$ and each $j \in [\ell]$, $\mathbf{R}_{i,j} \in R^{m\times n}$ and $\mathbf{L}_{i,j} \in R^{n\times m}$ are uniformly random matrices subject to the restriction that for each $i \in [Z-1]$ and for each $j, j' \in [\ell]$ such that $j \neq j'$, we have

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

  for some block-diagonal matrix $\mathbf{D}_{i,j} \in R^{m\times m}$.

- *A set of terms of the form $\{\widetilde{\mathbf{B}}_{t'}\}$ where each $\widetilde{\mathbf{B}}_{t'}$ is an n-"blockwise-random" matrix in $R^{n(L-\ell)\times n(L-\ell)}$,*
- *Permutations $\pi_0, \pi_1 \in S_L$ (where S_L denotes the symmetric group of L elements).*

outputs a set of terms of the form $\{\mathbf{M}_{i,\mathbf{B}_t,\widetilde{\mathbf{B}}_{t'},\pi_0,\pi_1}\}_{i\in[Z],t\in[T],t'\in[T']}$, where each $\mathbf{M}_{i,\mathbf{B}_t,\widetilde{\mathbf{B}}_{t'},\pi_0,\pi_1}$ is a matrix of the form

$$\mathbf{M}_{i,\mathbf{B}_t,\widetilde{\mathbf{B}}_{t'}} = \widetilde{\mathbf{L}}_i\bar{\pi}_0 \begin{bmatrix} \mathbf{B}_t & \mathbf{0}_R \\ \\ \mathbf{0}_R & \widetilde{\mathbf{B}}_{t'} \end{bmatrix} \bar{\pi}_1\widetilde{\mathbf{R}}_i,$$

with $\widetilde{\mathbf{L}}_i$ and $\widetilde{\mathbf{R}}_i$ for $i \in [Z]$ also distributed as described above, i.e., we have

$$\widetilde{\mathbf{R}}_i = \begin{bmatrix}\widetilde{\mathbf{R}}_{i,1} & \dots & \widetilde{\mathbf{R}}_{i,\ell}\end{bmatrix}, \quad \widetilde{\mathbf{L}}_i = \begin{bmatrix}\widetilde{\mathbf{L}}_{i,1} \\ \vdots \\ \widetilde{\mathbf{L}}_{i,\ell}\end{bmatrix},$$

where for each $i \in [Z]$ and each $j \in [\ell]$, $\widetilde{\mathbf{R}}_{i,j} \in R^{m'\times n}$ and $\widetilde{\mathbf{L}}_{i,j} \in R^{n\times m'}$ are uniformly random matrices subject to the restriction that for each $i \in [Z-1]$ and for each $j, j' \in [\ell]$ such that $j \neq j'$, we have

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

for some block-diagonal matrix $\mathbf{D}_{i,j} \in R^{m'\times m'}$, and $\bar{\pi}_0 \in R^{nL\times nL}$ and $\bar{\pi}_1 \in R^{nL\times nL}$ are the n-"block-wise" R-embedded matrix representations of π_0 and π_1, respectively.

Proof. We again prove this by showing the existence of such an algorithm. The algorithm proceeds through the following steps:

1. For each $i \in [Z]$ and each $j \in [\ell+1, L]$, sample additional matrices $\mathbf{R}_{i,j} \in R^{m \times n}$ and $\mathbf{L}_{i,j} \in R^{n \times m}$ subject to the restriction that for each $i \in [Z-1]$ and for each $j, j' \in [\ell+1, L]$ such that $j \neq j'$, we have

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

for some uniformly random block-diagonal matrix $\mathbf{D}_{i,j} \in R^{m \times m}$. Note that this can be done efficiently using the same procedure as sampling the guard matrices for our iaiO construction in Section 4.

2. Set the following (hypothetically) for each $i \in [Z]$:

$$\widehat{\mathbf{R}}_i = \begin{bmatrix}\mathbf{R}_{i,1} & \dots & \mathbf{R}_{i,\ell} & \mathbf{R}_{i,\ell+1} & \dots & \mathbf{R}_{i,L}\end{bmatrix} (\bar{\pi}_0)^{-1}, \quad \widehat{\mathbf{L}}_i = (\bar{\pi}_1)^{-1} \begin{bmatrix}\mathbf{L}_{i,1} \\ \vdots \\ \mathbf{L}_{i,\ell} \\ \mathbf{L}_{i,\ell+1} \\ \vdots \\ \mathbf{L}_{i,L}\end{bmatrix},$$

where $(\bar{\pi}_0)^{-1} \in R^{nL \times nL}$ and $(\bar{\pi}_1)^{-1} \in R^{nL \times nL}$ are the n-"block-wise" R-embedded matrix representations of $(\pi_0)^{-1}$ and $(\pi_1)^{-1}$, respectively.

3. For any $i \in [Z], t \in [T], t' \in [T']$, given $\mathbf{M}_{i,\mathbf{B}_t} = \mathbf{L}_i\mathbf{B}_t\mathbf{R}_i$, simulate

$$\widehat{\mathbf{M}}_{i,\mathbf{B}_t,\widetilde{\mathbf{B}}_{t'}} := \widehat{\mathbf{L}}_i\bar{\pi}_0 \begin{bmatrix}\mathbf{B}_t & \mathbf{0}_R \\ \mathbf{0}_R & \widetilde{\mathbf{B}}_{t'}\end{bmatrix} \bar{\pi}_0\widehat{\mathbf{R}}_i$$

as:

$$\widehat{\mathbf{M}}_{i,\mathbf{B}_t,\widetilde{\mathbf{B}}_{t'}} = \mathbf{M}_{i,\mathbf{B}_t} + \begin{bmatrix}\mathbf{R}_{i,\ell+1} & \dots & \mathbf{R}_{i,L}\end{bmatrix} \widetilde{\mathbf{B}}_{t'} \begin{bmatrix}\mathbf{L}_{i,\ell+1} \\ \vdots \\ \mathbf{L}_{i,L}\end{bmatrix}.$$

4. For each $i \in [Z]$ and each $j \in [\ell+1, L]$, sample uniformly random matrices $\mathbf{R}^*_i \in R^{m' \times m}$ and $\mathbf{L}^*_i \in R^{m \times m'}$ subject to the restriction that for each $i \in [Z-1]$, we have

$$\mathbf{R}^*_i\mathbf{L}^*_{i+1} = \mathbf{D}^*_i,$$

for some uniformly random block-diagonal matrix $\mathbf{D}^*_i \in R^{m' \times m'}$. This is again done in exactly the same way as sampling the guard matrices for our iaiO construction in Section 4.

5. Set the following (hypothetically) for each $i \in [Z]$:

$$\widetilde{\mathbf{R}}_i = \widehat{\mathbf{R}}_i\mathbf{R}^*_i, \quad \widetilde{\mathbf{L}}_i = \mathbf{L}^*_i\widehat{\mathbf{L}}_i,$$

where $\widehat{\mathbf{R}}_i$ and $\widehat{\mathbf{L}}_i$ are as set (hypothetically) in Step-3.

6. For any $i \in [Z], t \in [T], t' \in [T']$, given $\mathbf{M}_{i,\mathbf{B}_t} = \mathbf{L}_i\mathbf{B}_t\mathbf{R}_i$, simulate

$$\widetilde{\mathbf{M}}_{i,\mathbf{B}_t,\widetilde{\mathbf{B}}_{t'}} := \widetilde{\mathbf{L}}_i\bar{\pi}_0 \begin{bmatrix}\mathbf{B}_t & \mathbf{0}_R \\ \mathbf{0}_R & \widetilde{\mathbf{B}}_{t'}\end{bmatrix} \bar{\pi}_1\widetilde{\mathbf{R}}_i$$

as:

$$\widetilde{\mathbf{M}}_{i,\mathbf{B}_t,\widetilde{\mathbf{B}}_{t'}} = \mathbf{L}^*_i\widehat{\mathbf{M}}_{i,\mathbf{B}_t,\widetilde{\mathbf{B}}_{t'}}\mathbf{R}^*_i = \mathbf{L}^*_i \left(\mathbf{M}_{i,\mathbf{B}_t} + \begin{bmatrix}\mathbf{R}_{i,\ell+1} & \dots & \mathbf{R}_{i,L}\end{bmatrix} \widetilde{\mathbf{B}}_{t'} \begin{bmatrix}\mathbf{L}_{i,\ell+1} \\ \vdots \\ \mathbf{L}_{i,L}\end{bmatrix} \right) \mathbf{R}^*_i.$$

We can again use arguments similar to those in the proof of the simple lifting lemma to show that the simulated "mega-guards" of the form $\widetilde{\mathbf{R}}_i = \widehat{\mathbf{R}}_i \mathbf{R}_i^*$ and $\widetilde{\mathbf{L}}_i = \mathbf{L}_i^* \widehat{\mathbf{L}}_i$ are distributed appropriately. In particular, the only difference between the simple and the generalized lemma is that the latter uses "permuted" versions of the mega-guard matrices; however, it is easy to see that the permutations do not affect the products of the simulated "mega-guards"; in other words, they remain uniformly random subject to the constraints defined by their product distributions. This completes the proof of Lemma 7.2. □

7.2 From "Baby" Subspace Hiding Assumption To Full Assumption

In this section, we state and prove the following lemmas.

Lemma 7.3. *Assuming that the baby subspace hiding assumption-1 as in Definition 5.3 holds over a ring R for parameters (m, L, N) for some $m, L, N = \mathrm{poly}(\lambda)$, the subspace hiding assumption-1 as in Definition 5.1 holds over the same ring R for parameters (m, L, N, T, Z) for any $T, Z = \mathrm{poly}(\lambda)$.*

Lemma 7.4. *Assuming that the baby subspace hiding assumption-2 as in Definition 5.4 holds over a ring R for parameters (m, L, N) for some $m, L, N = \mathrm{poly}(\lambda)$, the subspace hiding assumption-2 as in Definition 5.2 holds over the same ring R for parameters (m, L, N, T, Z) for any $T, Z = \mathrm{poly}(\lambda)$.*

Proof of Lemma 7.3. The proof follows from a direct application of the generalized version of the lifting lemma (Lemma 7.2) as described in Section 7.1. More concretely, to prove this lemma, we show that given a PPT algorithm $\mathcal{A}$ that breaks the general subspace hiding assumption-1 with non-negligible advantage, one can construct a PPT algorithm $\mathcal{B}$ that breaks the baby subspace hiding assumption-1 with non-negligible advantage.

Inputs to $\mathcal{B}$. The algorithm $\mathcal{B}$ receives as input the following matrices sampled from appropriate distributions (in what follows, all block diagonal matrices $\mathbf{D}$ are of dimension $10m \times 10m$, and a total of $(L + N)$ guard matrix pairs are used; also recall that the baby subspace hiding assumption-1 is a special case of the subspace hiding assumption-1 where there are only two permutation slots that are, by default, the challenge permutation slots, and no additional slots, i.e., we have $T = 2$ and $Z = 0$):

1. **Generators:**
   - For each $\ell \in [L], b \in \{0, 1\}$: many samples of the form $\mathbf{L}_\ell \mathbf{D}^{(1, N_{t,\ell,b})} \mathbf{R}_\ell$, where $\mathbf{D}^{(t, N_{1,\ell,b})}$ is a permutation-embedded block-diagonal matrix sampled from the appropriate distribution.
   - For each $i \in [N]$: many samples of the form $\mathbf{L}_{L+i} \mathbf{D}^{(1,I)} \mathbf{R}_{L+i}$, where $I \in \{0, 1\}^{5\times 5}$ denotes the identity permutation, and $\mathbf{D}^{(1,I)}$ is an identity-permutation-embedded block-diagonal matrix sampled from the appropriate distribution.

   In each case, the "end guards" $\mathbf{L}_1$ and $\mathbf{R}_{L+N}$ are specially structured, as described in Section 4.3.
2. **Challenge-Relevant Elements**:
   - For each $\ell \in [L]$ and for each $b \in \{0, 1\}$: many samples of the form
   $$\mathbf{L}_\ell \mathbf{D}^{(1, N_{1,\ell,b}), (2, N_{2,\ell,b})} \mathbf{R}_\ell,$$
   where $\mathbf{D}^{(1, N_{1,\ell,b}), (2, N_{2,\ell,b})}$ is a permutation-embedded block-diagonal matrix sampled from the appropriate distribution.
   - For each $i \in [N]$: many samples of the form
   $$\mathbf{L}_{L+i} \mathbf{D}^{(1,I), (2,I)} \mathbf{R}_{L+i},$$
   where $I \in \{0, 1\}^{5\times 5}$ denotes the identity permutation, and $\mathbf{D}^{(1,I), (2,I)}$ is an identity-permutation-embedded block-diagonal matrix sampled from the appropriate distribution.

Once again, in each case, the "end guards" $\mathbf{L}_1$ and $\mathbf{R}_{L+N}$ are specially structured, as described in Section 4.3.

3. **Special Challenge-Index Elements**: Many samples of the form

$$\mathbf{L}_{(L+i^*)}\mathbf{D}^{(2,I)}\mathbf{R}_{(L+i^*)},$$

where $I \in \{0,1\}^{5\times 5}$ again denotes the identity permutation, and $\mathbf{D}^{(2,I)}$ is an identity-permutation-embedded block-diagonal matrix sampled from the appropriate distribution.

Also, in the case when $i^* = N$, the right "end guard" $\mathbf{R}_{L+N}$ is specially structured, as described in Section 4.3.

4. **The Challenge Element**: A sample that has one of the two following forms:

- $\mathbf{L}_{(L+i^*)}\mathbf{D}^{(1,I)}\mathbf{R}_{(L+i^*)}$ if $b = 0$, or
- $\mathbf{L}_{(L+i^*)}\mathbf{D}^{(1,I),(2,I)}\mathbf{R}_{(L+i^*)}$ if $b = 1$,

where $I \in \{0,1\}^{5\times 5}$ again denotes the identity permutation, and $\mathbf{D}^{(1,I)}$ and $\mathbf{D}^{(1,I),(2,I)}$ are block-diagonal matrices sampled from the appropriate distribution.

Also, in the case when $i^* = N$, the right "end guard" $\mathbf{R}_{L+N}$ is specially structured, as described in Section 4.3.

Inputs to $\mathcal{A}$ (to be simulated by $\mathcal{B}$). The algorithm $\mathcal{B}$ in turn needs to simulate and provide to the adversary $\mathcal{A}$ in the general subspace hiding assumption game with the following matrices sampled from appropriate distributions (in what follows, all block diagonal matrices $\widetilde{\mathbf{D}}$ are of dimension $5m'(T+Z) \times 5m'(T+Z)$, and a total of $(L+N)$ guard matrix pairs are used):

1. **Generators:**

- For each $\ell \in [L]$, $b \in \{0,1\}$ and each $t \in [T] \setminus \{t_1^*\}$: many samples of the form $\widetilde{\mathbf{L}}_\ell \widetilde{\mathbf{D}}^{(t,N_{t,\ell,b})}\widetilde{\mathbf{R}}_\ell$, where $\widetilde{\mathbf{D}}^{(t,N_{t,\ell,b})}$ is a permutation-embedded block-diagonal matrix sampled from the appropriate distribution.
- For each $i \in [N]$ and each $t \in [T]\setminus\{t_1^*\}$: many samples of the form $\widetilde{\mathbf{L}}_{L+i}\widetilde{\mathbf{D}}^{(t,I)}\widetilde{\mathbf{R}}_{L+i}$, where $I \in \{0,1\}^{5\times 5}$ denotes the identity permutation, and $\widetilde{\mathbf{D}}^{(t,I)}$ is an identity-permutation-embedded block-diagonal matrix sampled from the appropriate distribution.
- For each $\ell \in [L+N]$ and each $z \in [Z]$: many samples of the form $\widetilde{\mathbf{L}}_\ell \widetilde{\mathbf{D}}^{(T+z)}\widetilde{\mathbf{R}}_\ell$, where $\widetilde{\mathbf{D}}^{(T+z)}$ is a block-diagonal matrix sampled from the appropriate distribution.

In each case, the "end guards" $\widetilde{\mathbf{L}}_1$ and $\widetilde{\mathbf{R}}_{L+N}$ are specially structured, as described in Section 4.3.

2. **Challenge-Relevant Elements**:

- For each $\ell \in [L]$ and for each $b \in \{0,1\}$: many samples of the form

$$\widetilde{\mathbf{L}}_\ell \widetilde{\mathbf{D}}^{(t_0^*,N_{t_0^*,\ell,b}),(t_1^*,N_{t_1^*,\ell,b})}\widetilde{\mathbf{R}}_\ell,$$

where $\widetilde{\mathbf{D}}^{(t_0^*,N_{t_0^*,\ell,b}),(t_1^*,N_{t_1^*,\ell,b})}$ is a permutation-embedded block-diagonal matrix sampled from the appropriate distribution.

- For each $i \in [N]$: many samples of the form

$$\widetilde{\mathbf{L}}_{L+i}\widetilde{\mathbf{D}}^{(t_0^*,I),(t_1^*,I)}\widetilde{\mathbf{R}}_{L+i},$$

where $I \in \{0,1\}^{5\times 5}$ denotes the identity permutation, and $\widetilde{\mathbf{D}}^{(t_0^*,I),(t_1^*,I)}$ is an identity-permutation-embedded block-diagonal matrix sampled from the appropriate distribution.

Once again, in each case, the "end guards" $\widetilde{\mathbf{L}}_1$ and $\widetilde{\mathbf{R}}_{L+N}$ are specially structured, as described in Section 4.3.

3. **Special Challenge-Index Elements**: Many samples of the form

$$\widetilde{\mathbf{L}}_{(L+i^*)}\widetilde{\mathbf{D}}^{(t_1^*,I)}\widetilde{\mathbf{R}}_{(L+i^*)},$$

where $I \in \{0,1\}^{5\times 5}$ again denotes the identity permutation, and $\widetilde{\mathbf{D}}^{(t_1^*,I)}$ is an identity-permutation-embedded block-diagonal matrix sampled from the appropriate distribution.

Also, in the case when $i^* = N$, the right "end guard" $\widetilde{\mathbf{R}}_{L+N}$ is specially structured, as described in Section 4.3.

4. **The Challenge Element**: A sample that has one of the two following forms:

- $\widetilde{\mathbf{L}}_{(L+i^*)}\widetilde{\mathbf{D}}^{(t_0^*,I)}\widetilde{\mathbf{R}}_{(L+i^*)}$ if $b = 0$, or
- $\widetilde{\mathbf{L}}_{(L+i^*)}\widetilde{\mathbf{D}}^{(t_0^*,I),(t_1^*,I)}\widetilde{\mathbf{R}}_{(L+i^*)}$ if $b = 1$,

where $I \in \{0,1\}^{5\times 5}$ again denotes the identity permutation, and $\widetilde{\mathbf{D}}^{(t_0^*,I)}$ and $\widetilde{\mathbf{D}}^{(t_0^*,I),(t_1^*,I)}$ are block-diagonal matrices sampled from the appropriate distribution.

Also, in the case when $i^* = N$, the right "end guard" $\widetilde{\mathbf{R}}_{L+N}$ is specially structured, as described in Section 4.3.

We argue that a simple application of the generalized version of the lifting lemma in Section 7.1 allows us to argue that $\mathcal{B}$ can simulate the inputs to $\mathcal{A}$ from its own inputs. More specifically, we require four applications of the lemma - one for each class of terms (generators, challenge-relevant elements, special challenge-index elements, and the challenge element itself) that $\mathcal{B}$ needs to provide to $\mathcal{A}$ from the corresponding class of terms it receives as input, with the permutation matrices in the generalized version of the lifting lemma programmed to map the challenge indices in the baby subspace hiding assumption-1 to the challenge indices (t_0^*, t_1^*) in the full/general subspace assumption-1.

We illustrate this in details for the special case where $(t_0^*, t_1^*) = (1, 2)$. To begin with, note that all $\mathbf{D}$ and $\widetilde{\mathbf{D}}$ matrices in the aforementioned set of terms (i.e., inputs to $\mathcal{B}$ and inputs to $\mathcal{A}$) are all $5m$-blockwise random. In addition, observe the following:

1. **Generator Elements:** To begin with, we have:

$$\widetilde{\mathbf{D}}^{(t,N_{t,\ell,b})} = \begin{cases} \begin{bmatrix} \mathbf{D}^{(1,N_{t,\ell,b})} & \mathbf{0}_R \\ \mathbf{0}_R & \mathbf{0}_R \end{bmatrix} & \text{if t = 1,} \\ \begin{bmatrix} \mathbf{0}_R & \mathbf{0}_R \\ \mathbf{0}_R & \widetilde{\widetilde{\mathbf{D}}}^{(t,N_{t,\ell,b})} \end{bmatrix} & \text{otherwise for approproately formatted } \widetilde{\widetilde{\mathbf{D}}}^{(t,N_{t,\ell,b})}. \end{cases}$$

Also, we have

$$\widetilde{\mathbf{D}}^{(t,I)} = \begin{cases} \begin{bmatrix} \mathbf{D}^{(1,I)} & \mathbf{0}_R \\ \mathbf{0}_R & \mathbf{0}_R \end{bmatrix} & \text{if t = 1,} \\ \begin{bmatrix} \mathbf{0}_R & \mathbf{0}_R \\ \mathbf{0}_R & \widetilde{\widetilde{\mathbf{D}}}^{(t,I)} \end{bmatrix} & \text{otherwise for approproately formatted } \widetilde{\widetilde{\mathbf{D}}}^{(t,I)}. \end{cases}$$

Finally, we have

$$\widetilde{\mathbf{D}}^{(T+z)} = \begin{bmatrix} \mathbf{0}_R & \mathbf{0}_R \\ \mathbf{0}_R & \widetilde{\widetilde{\mathbf{D}}}^{(T+z)} \end{bmatrix} \text{ for approproately formatted } \widetilde{\widetilde{\mathbf{D}}}^{(T+z)}.$$

A simple application of the special lifting lemma (Lemma 7.1) in Section 7.1 allows us argue that $\mathcal{B}$ can simulate the generator elements to be provided to $\mathcal{A}$ given its own input generator elements.

2. **Challenge-Relevant Elements:** Next, observe the following: when $(t_0^*, t_1^*) = (1, 2)$, we have

$$\widetilde{\mathbf{D}}^{(t_0^*, N_{t_0^*,\ell,b}),(t_1^*, N_{t_1^*,\ell,b})} = \begin{bmatrix} \mathbf{D}^{(1,N_{1,\ell,b}),(2,N_{2,\ell,b})} & \mathbf{0}_R \\ \mathbf{0}_R & \mathbf{0}_R \end{bmatrix}.$$

Also, we have

$$\widetilde{\mathbf{D}}^{(t_0^*, I),(t_1^*, I)} = \begin{bmatrix} \mathbf{D}^{(1,I),(2,I)} & \mathbf{0}_R \\ \mathbf{0}_R & \mathbf{0}_R \end{bmatrix}$$

Again, a straightforward application of the special lifting lemma (Lemma 7.1) in Section 7.1 allows us argue that $\mathcal{B}$ can simulate the challenge-relevant elements to be provided to $\mathcal{A}$ given its own input challenge-relevant elements.

3. **Special Challenge-Index Elements:** Further, observe the following: when $(t_0^*, t_1^*) = (1, 2)$, we have

$$\widetilde{\mathbf{D}}^{(t_1^*, I)} = \begin{bmatrix} \mathbf{D}^{(2,I)} & \mathbf{0}_R \\ \mathbf{0}_R & \mathbf{0}_R \end{bmatrix}.$$

Again, a straightforward application of the special lifting lemma (Lemma 7.1) in Section 7.1 allows us argue that $\mathcal{B}$ can simulate the special-challenge-index elements to be provided to $\mathcal{A}$ given its own input challenge-relevant elements.

4. **The Challenge Element:** Finally, observe the following:

$$\widetilde{\mathbf{D}}^{(t_0^*, I)} = \begin{bmatrix} \mathbf{D}^{(1,I)} & \mathbf{0}_R \\ \mathbf{0}_R & \mathbf{0}_R \end{bmatrix}, \quad \widetilde{\mathbf{D}}^{(t_0^*, I),(t_1^*, I)} = \begin{bmatrix} \mathbf{D}^{(1,I),(2,I)} & \mathbf{0}_R \\ \mathbf{0}_R & \mathbf{0}_R \end{bmatrix}.$$

Yet again, a straightforward application of the special lifting lemma (Lemma 7.1) in Section 7.1 allows us argue that $\mathcal{B}$ can simulate the challenge element to be provided to $\mathcal{A}$ given its own input challenge element.

Finally, to generalize the argument to any arbitrary pair of challenge permutation slots (t_0^*, t_1^*), we resort to the generalized lifting lemma (Lemma 7.2) in Section 7.1 that additionally allows the internal matrices in the output distribution to be permuted along the rows and/or columns. In particular, the simulation strategy involves $\mathcal{B}$ additionally using a ring-embedded permutation matrix $\bar{pi}_0$ corresponding to the permutation map $\pi_0 : S_{T+Z} \to S_{T+Z}$ (where S_{T+Z} denotes the symmetric group of $(T + Z)$ elements) that "swaps" the challenge permutation slots t_0^* and t_1^* with the challenge permutation slots 1 and 2, respectively, i.e. for any $x \in [T + Z]$, we have:

$$\pi_0(x) = \begin{cases} t_0^* & \text{if } x = 1, \\ t_1^* & \text{if } x = 2, \\ 1 & \text{if } x = t_0^*, \\ 2 & \text{if } x = t_1^*. \end{cases}$$

The rest of the simulation remains unchanged. This completes the proof of Lemma 7.3.

Proof of Lemma 7.4. The proof of Lemma 7.4 follows analogously and is hence not detailed separately.

7.3 An Auxiliary Subspace Hiding Assumption

In this section we define an auxiliary subspace hiding assumption, and its corresponding baby version. It can be seen as a simplified version of the subspace hiding assumptions introduced in Section 5. We introduce this auxiliary assumption and prove its security; more specifically, we prove security of the baby version of this assumption (one can prove the full assumption from the baby version by a similar application of the lifting lemma (Lemma 7.2) as shown in

Section 7.2). We then use this assumption as a stepping stone to prove the security of our main baby subspace hiding assumptions, namely the permutation baby subspace hiding assumption (Definition 5.3) and the half permutation baby subspace hiding assumption (Definition 5.4).

In the rest of the section, we use similar notation for block-diagonal matrices $\mathbf{D}_i$ and a similar structure for the "guard" matrices $(\mathbf{L}_i, \mathbf{R}_i)$ as in Section 5, albeit with some differences in the dimensions of the sub-matrices, as laid out below.

Diagonal Matrices. We again consider a class of block diagonal matrices $\mathbf{D} \in R^{(M)\times(M)}$; however, instead of setting $M = 5mL$, we now have

$$M = \sum_{\ell=1}^{L} m_\ell.$$

We again break down $\mathbf{D}$ into an L by L blockwise structure: in other words, we assume the $\mathbf{D}$ is composed of L square diagonal blocks $\{\mathbf{D}_\ell\}_{\ell\in[L]}$, where the ℓth block is of size $m_\ell \times m_\ell$, unlike in Section 5, where each block was of size $5m \times 5m$. Graphically, this looks like

$$\mathbf{D} = \begin{bmatrix} \mathbf{D}_1 & & & & \\ & \mathbf{D}_2 & & \mathbf{0}_R & \\ & & \mathbf{D}_3 & & \\ & & & \ddots & \\ & \mathbf{0}_R & & & \\ & & & \ddots & \\ & & & & \mathbf{D}_L \end{bmatrix},$$

where for each $\ell \in [L]$, we have $\mathbf{D}_\ell \in R^{m_\ell \times m_\ell}$. Modulo these modifications surrounding variable-dimension sub-matrices, we use the same notation around diagonal matrices as in Section 5. More concretely, we have:

1. $\mathbf{D}^i \in R^{M\times M}$ denotes a block diagonal matrix with the restriction that *all diagonal blocks except the* i^{th} *diagonal block* are zero, while the ith diagonal block is a uniformly random matrix in $R^{m_i \times m_i}$.
2. $\mathbf{D}^{i,j} \in R^{M\times M}$ denotes a block diagonal matrix with the restriction that *all diagonal blocks except the* i^{th} *and* j^{th} *diagonal blocks* are zero, while the ith and j^{th} diagonal blocks are uniformly random matrices in $R^{m_i \times m_i}$ and $R^{m_j \times m_j}$, respectively.

The Guard Matrices. We will again use guard matrices as in the subspace hiding assumptions in Section 5, albeit with similar variable-dimension sub-matrices as in the diagonal matrices. Let n, L and $m_1, \ldots, m_L$ be integers such that for each $\ell \in [L]$, we have $m_\ell \ll n$. Also, for each $j \in [L]$, let

$$M_j = \sum_{\ell=1}^{j} m_\ell,$$

and let $M := M_L$. We use a set of "left guard" matrices $\mathbf{L}_i \in R^{n\times M}$ for $i \in [2, N]$, and a set of "right guard" matrices $\mathbf{R}_i \in R^{M\times n}$ for $i \in [2, N]$. In addition, we consider sub-matrices of $\mathbf{L}_i$ and $\mathbf{R}_i$, respectively. Also:

- Let $\mathbf{L}_{i,j} \in R^{n\times m_j}$ be the sub-matrix of $\mathbf{L}_i$ consisting of the $(M_{j-1}+1)$-th through the M_j-th *columns* of $\mathbf{L}_i$.
- Let $\mathbf{R}_{i,j} \in R^{m_j\times n}$ be the sub-matrix of $\mathbf{R}_i$ consisting of the $(M_{j-1}+1)$-th through the M_j-th *rows* of $\mathbf{R}_i$.

In other words, our matrices have the following structure:

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

Experiment $\mathsf{Expt}^{\text{Aux-Subspace-Hiding}}_{R,(m_1,\ldots,m_L),N,b}$:

1. The adversary $\mathcal{A}$ chooses two challenge-slots $i_0^*, i_1^* \in [L]$ and a challenge-index $j^* \in [N]$. It provides (i_0^*, i_1^*, j^*) to the challenger.
2. The challenger provides the adversary with the following matrices sampled from appropriate distributions (in what follows, all block diagonal matrices $\mathbf{D}$ are "variable-dimension" with overall dimension $M \times M$ for $M = \sum_{\ell=1}^{L} m_\ell$, and a total of N guard matrix pairs are used):
   (a) **Generators:** For each $j \in [N]$ and each $i \in [L] \setminus \{i_1^*\}$: many samples of the form $\mathbf{L}_j \mathbf{D}^i \mathbf{R}_j$, where $\mathbf{D}^i$ is a randomly sampled matrix of the appropriate form as described above. For $j = 1$ and $j = N$, the "end guards" $\mathbf{L}_1$ and $\mathbf{R}_N$ are specially structured, as described above.
   (b) **Challenge-Relevant Elements**: For each $j \in [N]$: many samples of the form $\mathbf{L}_j \mathbf{D}^{i_0^*, i_1^*} \mathbf{R}_j$, where $\mathbf{D}^{i_0^*, i_1^*}$ is a randomly sampled matrix of the appropriate form as described above. For $j = 1$ and $j = N$, the "end guards" $\mathbf{L}_1$ and $\mathbf{R}_N$ are specially structured, as described above.
   (c) **Special Challenge-Index Elements**: Many samples of the form $\mathbf{L}_{j^*} \mathbf{D}^{i_1^*} \mathbf{R}_{j^*}$, where $\mathbf{D}^{i_1^*}$ is a randomly sampled matrix of the appropriate form as described above. In the case where $j^* = 1$ or $j^* = N$, the "end guards" $\mathbf{L}_1$ and $\mathbf{R}_N$ are specially structured, as described above.
   (d) **The Challenge Element**: A sample that is of one of the two following forms:
   - $\mathbf{L}_{j^*} \mathbf{D}^{i_0^*} \mathbf{R}_{j^*}$ if $b = 0$, or
   - $\mathbf{L}_{j^*} \mathbf{D}^{i_0^*, i_1^*} \mathbf{R}_{j^*}$ if $b = 1$,

   where $\mathbf{D}^{i_0^*}$ and $\mathbf{D}^{i_0^*, i_1^*}$ are randomly sampled matrices of the appropriate form as described above. In the case where $j^* = 1$ or $j^* = N$, the "end guards" $\mathbf{L}_1$ and $\mathbf{R}_N$ are specially structured, as described above.

Figure 15: The Auxiliary 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

$$\mathbf{R}_{i,j} \cdot \mathbf{L}_{i+1,j'} = \mathbf{0}_R \in R^{m_j \times m_{j'}}.$$

In other words, the product $\mathbf{R}_i \cdot \mathbf{L}_{i+1} \in R^{M \times M}$ is required to have the form of a "variable-dimension" block diagonal matrix $\mathbf{D}$ as we defined above. Finally, the "end guards" $\mathbf{L}_1$ and $\mathbf{R}_N$ are specially structured in the sense that we have:

$$\mathbf{L}_1 = \begin{bmatrix} \mathbf{I}_1 & \mathbf{I}_2 & \ldots & \mathbf{I}_L \end{bmatrix}, \quad \mathbf{R}_N = \begin{bmatrix} \mathbf{I}_1 \\ \mathbf{I}_2 \\ \vdots \\ \mathbf{I}_L \end{bmatrix},$$

where for each $\ell \in [L]$, the matrix $\mathbf{I}_\ell \in R^{m_\ell \times m_\ell}$ is a ring-embedding of the identity matrix with dimension $(m_\ell \times m_\ell)$.

Equipped with the aforementioned notations, we now state the subspace hiding assumption.

The Auxiliary Subspace Hiding Assumption. For any ring R, any choice of parameters $L, N, (m_1, \ldots, m_L)$ and each $b \in \{0, 1\}$, let $\mathsf{Expt}^{\text{Aux-Subspace-Hiding}}_{R,(m_1,\ldots,m_L),N,b}$ denote an experiment between a challenger and an adversary $\mathcal{A}$ as defined in Figure 15.

Definition 7.5. (Auxiliary Subspace Hiding Assumption.) The auxiliary subspace hiding assumption is said to hold over a ring R if for any security parameter λ, any choice of parameters $L, N, m_1, \ldots, m_L = \mathrm{poly}(\lambda)$, and for any PPT adversary $\mathcal{A}$, the views of the adversary $\mathcal{A}$ in the experiments $\mathsf{Expt}^{\text{Aux-Subspace-Hiding}}_{R,(m_1,\ldots,m_L),N,0}$ and $\mathsf{Expt}^{\text{Aux-Subspace-Hiding}}_{R,(m_1,\ldots,m_L),N,1}$ are computationally indistinguishable.

The Auxiliary "Baby" Subspace Hiding Assumption. We also define a special case of this assumption, called the auxiliary "baby" subspace hiding assumption, for a fixed choice of parameter $L = 2$ with two unique dimension parameters (m_1, m_2).

Definition 7.6. (Auxiliary Baby Subspace Hiding Assumption.) The auxiliary baby subspace hiding assumption is said to hold over a ring R if for any security parameter λ, any choice of parameters $N, m_1, m_2 = \mathrm{poly}(\lambda)$ and for any PPT adversary $\mathcal{A}$, the views of the adversary $\mathcal{A}$ in the experiments $\mathsf{Expt}^{\text{Aux-Subspace-Hiding}}_{R,(m_1,m_2),N,0}$ and $\mathsf{Expt}^{\text{Aux-Subspace-Hiding}}_{R,(m_1,m_2),N,1}$ are computationally indistinguishable.

Note that in the auxiliary baby subspace hiding assumption, the challenge slot-pair (i^*_0, i^*_1) is $(1, 2)$ by default.

We next prove the auxiliary baby subspace assumption in Section 7.3. Using this proof as a stepping stone, we prove the permutation baby subspace hiding assumption (Definition 5.3) and the half permutation baby subspace hiding assumption (Definition 5.4) in Sections 7.4 and 7.5, respectively.

7.3.1 Overview of Proof of Auxiliary Baby Subspace Hiding Assumption

We now prove that the auxiliary baby subspace hiding assumption as in Definition 7.6 holds over any generic ring R such that for any $m, Q = \mathrm{poly}(\lambda)$ where m is any arbitrary positive integer satisfying $m > 3\lambda$ and Q is any arbitrary positive integer, we have

$$(\mathbf{A}, \mathbf{AK}) \overset{c}{\approx} (\mathbf{A}, \mathbf{U}), \quad (\mathbf{B}, \mathbf{K'B}) \overset{c}{\approx} (\mathbf{B}, \mathbf{V})$$

where $\mathbf{K}, \mathbf{K'} \leftarrow R^{m\times m}$, $\mathbf{A}, \mathbf{U} \leftarrow R^{Qm\times m}$, and $\mathbf{B}, \mathbf{V} \leftarrow R^{m\times Qm}$. While we do not use this assumption directly in our iaiO proof, looking ahead, we use it as a step towards proving the permutation and half permutation baby subspace hiding assumptions. Additionally, the intuition as to why this assumption holds true is essentially the intuition for why our iaiO scheme scheme is secure, so we go over the proof here in detail.

Before presenting the formal proof, we provide some additional insight into the auxiliary baby subspace hiding assumption and the proof strategy for the same using an example. Consider an instance of the auxiliary baby subspace hiding assumption experiment with $N = 4$, 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 sub-matrices of the appropriate dimensions m_1 and m_2, respectively; also, the guard sub-matrices are parameterized by the dimensions m_1 and m_2 as discussed in Section 5):

1. **Generators**:

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

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

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

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

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

2. **Challenge-Relevant Elements.**

The challenger provides the adversary $\mathcal{A}$ with several challenge-relevant elements of the following form (note that in the auxiliary baby subspace hiding assumption experiment, the challenge-slot pair is $(i_0^*, i_1^*) = (1, 2)$ by default):

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

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

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

$$\mathbf{H}_4 = \begin{bmatrix} | & | \\ \mathbf{L}_{4,1} & \mathbf{L}_{3,2} \\ | & | \end{bmatrix} \cdot \begin{bmatrix} \mathbf{X} \\ \mathbf{X} \end{bmatrix}$$

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 auxiliary baby subspace hiding 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 auxiliary baby subspace hiding assumption experiment by type and index. We label the generators as $\mathbf{G}_1, \ldots, \mathbf{G}_N$, the challenge-relevant elements as $\mathbf{H}_1, \ldots, \mathbf{H}_N$, 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 $j' \in [j^*+1, N]$, respectively:

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

The astute reader may observe at this point that these are the only relations that the adversary $\mathcal{A}$ can efficiently verify (all other product combinations will be indistinguishable from random). Thus, intuitively, we wish to establish that the terms provided by the challenger to the adversary $\mathcal{A}$ in the auxiliary baby subspace hiding 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 auxiliary baby subspace hiding assumption experiment. We revert back to our previous example where we considered an instance of the auxiliary baby subspace hiding assumption experiment with $N = 4$, 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 sub-matrices that *do not* need to satisfy constraints (i.e., sub-matrices 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 (note that for both the left and right guards, the first sub-matrix corresponds to the subspace with dimension m_1 and the other sub-matrix corresponds to the subspace with dimension m_2):

1. **Generators**:

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

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

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

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

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

2. **Challenge-Relevant Elements.**

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

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

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

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

$$\mathbf{H}_4 = \begin{bmatrix}| & | \\ \mathbf{L}_{4,1} & \mathbf{L}_{4,2} \\ | & |\end{bmatrix} \cdot \begin{bmatrix}\mathbf{X} \\ \mathbf{X}\end{bmatrix}$$

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

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

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 sub-matrices in the aforementioned depiction with uniformly random sub-matrices of the same dimension. This does not violate any of the aforementioned constraints as these are the sub-matrices for which additional orthogonal terms are *not* provided to the adversary $\mathcal{A}$. Additionally, once all the "blue" guard sub-matrices have been replaced with uniformly random sub-matrices of the same dimension, indistinguishability of the challenge element from a random matrix (independent of the challenge bit b) follows in a straightforward manner from Corollary 3.15.

Proof Strategy. Based on this observation, our proof strategy for the auxiliary baby subspace hiding assumption will be to replace the "blue" guard sub-matrices in the aforementioned depiction with uniformly random sub-matrices 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 Corollary 3.15, we can prove that the following are computationally indistinguishable whenever the ring R is the output ring of a ring-embedded 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.

Formally proving the auxiliary baby subspace hiding assumption requires a lot of applications of Theorem 3.13 and quite a bit of algebra. We have pushed most of the math to section A, which goes through a lot of mostly intuitive (but tedious) algebraic manipulations. The proof here should still be intuitive without all of these lemmas, though.

7.3.2 Formal Proof of Auxiliary Baby Subspace Hiding Assumption

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

Lemma 7.7. *The auxiliary baby subspace hiding assumption as in Definition 7.6 holds over any generic ring R such that for any $m, Q = \mathrm{poly}(\lambda)$ where m is any arbitrary positive integer satisfying $m > 3\lambda$ and Q is any arbitrary positive integer, we have*

$$(\mathbf{A}, \mathbf{AK}) \overset{c}{\approx} (\mathbf{A}, \mathbf{U}), \quad (\mathbf{B}, \mathbf{K}'\mathbf{B}) \overset{c}{\approx} (\mathbf{B}, \mathbf{V})$$

where $\mathbf{K}, \mathbf{K}' \leftarrow R^{m \times m}$, $\mathbf{A}, \mathbf{U} \leftarrow R^{Qm \times m}$, and $\mathbf{B}, \mathbf{V} \leftarrow R^{m \times Qm}$.

Proof Hybrids. We prove this lemma via a sequence of hybrid experiments $\mathcal{H}_0$ through $\mathcal{H}_N$, as described below, where $N = \mathrm{poly}(\lambda)$ parameterizes the auxiliary baby subspace hiding assumption (we also assume in the remainder of the description that $j^* \in [N]$ is the challenge-index chosen by the adversary $\mathcal{A}$ in the auxiliary baby subspace hiding assumption experiment, and that in all guard matrices, the first and second sub-matrices correspond to the subspaces with dimensions m_1 and m_2, respectively):

1. $\mathcal{H}_0$: This experiment is identical to the auxiliary baby subspace hiding assumption experiment with challenge element $\mathbf{C}_0$.

2. $\mathcal{H}_j$ for $j \in [1, j^* - 1]$: 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:

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

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

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

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

$$\mathbf{R}_{N-(j-j^*)-1} = \begin{bmatrix} -- & \mathbf{R}_{N-(j-j^*)-1,1} & -- \\ -- & \mathbf{R}_{N-(j-j^*)-1,2} & -- \end{bmatrix},$$

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

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

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

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

4. $\mathcal{H}_N$: In this hybrid, the challenger changes the structure of the challenge matrix. The challenger switches from $\mathbf{C}_0$ to $\mathbf{C}_1$, although it is important to emphasize that *the guard matrices are still randomized* so we do not technically have a proper baby subspace assumption challenge distribution.

5. $\mathcal{H}_j$ for $j \in [N+1, 2N-j^*]$: These hybrids are the "reverse" of hybrids $\mathcal{H}_j$ for $j \in [j^*, N-1]$. More precisely, experiment $\mathcal{H}_j$ is identical to the previous experiment $\mathcal{H}_{j-1}$ except for the manner in which the challenger generates the guard matrices $\mathbf{R}_{j-(N+1-j^*)}$ and $\mathbf{L}_{j-(N+1-j^*)+1}$. Suppose that in the experiment $\mathcal{H}_{j-1}$, the guard matrices $\mathbf{R}_{j-(N+1-j^*)}$ and $\mathbf{L}_{j-(N+1-j^*)}$ are generated as:

$$\mathbf{R}_{j-(N+1-j^*)} = \begin{bmatrix} -- & \mathbf{Y}_1 & -- \\ -- & \mathbf{R}_{j-(N+1-j^*),2} & -- \end{bmatrix},$$

$$\mathbf{L}_{j-(N+1-j^*)+1} = \begin{bmatrix} | & | \\ \mathbf{L}_{j-(N+1-j^*)+1,1} & \mathbf{Y}_2 \\ | & | \end{bmatrix},$$

for uniformly random $\mathbf{Y}_1, \mathbf{Y}_2$. In the experiment $\mathcal{H}_j$, the sub-matrices $\mathbf{Y}_1$ and $\mathbf{Y}_2$ are replaced with "real" guard matrices $\mathbf{R}_{j-(N+1-j^*),2}$ and $\mathbf{L}_{j-(N+1-j^*)+1,1}$. More concretely, in the experiment $\mathcal{H}_j$, we have

$$\mathbf{R}_{j-(N+1-j^*)} = \begin{bmatrix} -- & \mathbf{R}_{j-(N+1-j^*),1} & -- \\ -- & \mathbf{R}_{j-(N+1-j^*),2} & -- \end{bmatrix},$$

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

6. $\mathcal{H}_j$ for $j \in [2N-j^*+1, 2N-1]$: these hybrids are the "reverse" of hybrids $\mathcal{H}_j$ for $j \in [1, j^*-1]$. More precisely, experiment $\mathcal{H}_j$ is identical to the previous experiment $\mathcal{H}_{j-1}$ except for the manner in which the challenger generates the guard matrices $\mathbf{R}_{2N-j}$ and $\mathbf{L}_{2N-j+1}$. So, suppose that in the experiment $\mathcal{H}_{j-1}$, the guard matrices $\mathbf{R}_{2N-j}$ and $\mathbf{L}_{2N-j+1}$ are generated as:

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

where $\mathbf{X}_1$ and $\mathbf{X}_2$ are matrices of the appropriate dimensions with uniformly sampled entries from the ring R. In the experiment $\mathcal{H}_j$, the sub-matrices $\mathbf{X}_1$ and $\mathbf{X}_2$ are replaced with the correct guard matrices $\mathbf{R}_{2N-j,2}$ and $\mathbf{L}_{2N-j+1,1}$. More concretely, in the experiment $\mathcal{H}_j$, we have

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

Note that hybrid $2N-1$ is equivalent to the actual auxiliary baby subspace hiding assumption experiment with challenge element $\mathbf{C}_1$.

We also note that the numbering of the hybrids might be a little bit confusing, but the intuition is pretty straightforward: the hybrids start from the "left-most" matrices and move to the challenge index. Then, they jump to the "right-most" set of guards and move inwards (i.e. to the "left") until the challenge index is hit. Then, we switch the challenge term. Finally, we "undo" all of the randomizations we have made in the second half of our hybrids, essentially reversing our earlier hybrids in the exact reverse order.

Indistinguishability of Outer Hybrids. We start by proving that our "outer" hybrids–those that randomize the "outer" guards on the right and left hand sides–are true. We note that the four hybrids where we prove facts about outer guards ($\mathcal{H}_1$, $\mathcal{H}_{j^*}$, $\mathcal{H}_{2N-j^*}$, and $\mathcal{H}_{2N-1}$) all have essentially identical proofs, so we only prove the case for $\mathcal{H}_1$ in detail and define the rest as corollaries. We assume throughout that in all guard matrices and block-diagonal matrices, the first and second sub-matrices correspond to the subspaces with dimensions m_1 and m_2, respectively.

Lemma 7.8. *If R is a ring as in Lemma 7.7, then hybrid $\mathcal{H}_0$ is computationally indistinguishable from hybrid $\mathcal{H}_1$.*

Proof. In this lemma, our goal is to replace the guard sub-matrices

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

with those of the form

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

for uniformly random $\mathbf{X}_1$ and $\mathbf{X}_2$. We need to be able to make this swap and still simulate all of the other terms needed in the auxiliary baby subspace hiding assumption hybrid.

First, note that if we randomize $\mathbf{R}_{1,2}$ (i.e. swap it for a random $\mathbf{X}_1$) then $\mathbf{L}_{2,1}$ appears distributed uniformly at random (and thus identically to $\mathbf{X}_2$), since it no longer has any relationships with any other guards. So without loss of generality, we may assume that we only need to randomize $\mathbf{R}_{1,2}$ rather than both $\mathbf{R}_{1,2}$ and $\mathbf{L}_{2,1}$. We will use this simplification for the rest of this proof.

To start, we need to simulate the following terms in the hybrid slot:

1. **Generators**:

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

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

2. **Challenge-Relevant Elements.**

   The challenger provides the adversary $\mathcal{A}$ with several challenge-relevant elements of the following form:

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

The only terms in the auxiliary baby subspace hiding assumption that include the term we are switching in the hybrid are the challenge-relevant elements of the form $\mathbf{H}_1$.

Looking ahead, in Lemma A.4 (Appendix A), we state and prove the following: *let m, n, n_1 and n_2 be any integers such that*

$$m \geq 6(n + n_1 + n_2) \log |R|,$$

where R is a ring as in Lemma 7.7. Let the matrices $\mathbf{R}_1 \in R^{n_1 \times 2m}$, and $\mathbf{R}_2 \in R^{n_2 \times 2m}$ $\mathbf{L}_1 \in R^{2m \times n_1}$, $\mathbf{L}_2 \in R^{2m \times n_2}$, be random subject to the constraint that

$$\mathbf{R}_1 \mathbf{L}_2 = \mathbf{0}_R \quad \text{and} \quad \mathbf{R}_2 \mathbf{L}_1 = \mathbf{0}_R.$$

Suppose we sample sets of two matrices $\mathbf{X}_{1,i} \in R^{n \times n_1}$ and $\mathbf{X}_{2,i} \in R^{n \times n_2}$ uniformly at random for $i \in [N]$ (for some $N = \text{poly}(\lambda)$; λ being the security parameter), and let $\mathbf{Z}_i \in R^{n \times 2m}$ be matrices sampled uniformly at random as well for $i \in [N]$. We claim that, given the matrices:

$$\mathbf{R}_1, \mathbf{R}_2, \mathbf{L}_1, \mathbf{L}_2$$

samples of the following form are indistinguishable:

$$(\mathbf{X}_{1,i}\mathbf{R}_1 + \mathbf{X}_{2,i}\mathbf{R}_2) \text{ and } (\mathbf{Z}_i)$$

Also note that by Theorem 3.13, the following tuples are indistinguishable:

$$(\mathbf{X}_1, \mathbf{X}_{1,i} \cdot \mathbf{X}_1) \text{ and } (\mathbf{X}_1, \mathbf{Y}_{1,i}) \qquad \square$$

for each $\mathbf{Y}_{1,i} \in R^{n \times 2m}$ sampled uniformly at random.

So, given all of the terms provided by Lemma A.4 we can simulate all terms of the form $\mathbf{G}_1$, $\mathbf{G}_2$, and $\mathbf{H}_2$ directly. By Lemma A.4, we know that the honest simulation of the $\mathbf{H}_2$ terms as in $\mathcal{H}_0$ is indistinguishable from a simulation that replaces them with uniformly random terms. By our application of Theorem 3.13, we know that the simulation that replaces the $\mathbf{H}_2$ terms with random matrices $\mathbf{Y}_i$ is indistinguishable from the honest simulation of the $\mathbf{H}_2$ terms as in $\mathcal{H}_1$. This completes the proof of Lemma 7.8, as any adversary that can be used to distinguish these hybrids can break either Lemma A.4 or Theorem 3.13.

We also state the following corollaries, the proofs of which follow from essentially identical arguments to those in the proof of Lemma 7.8 and are hence not detailed. We note that the first and second corollaries here technically rely on Corollary A.7, which is just the "transpose" of Lemma A.5.

Corollary 7.9. *If R is a ring as in Lemma 7.7, then hybrid $\mathcal{H}_{j^*-1}$ is computationally indistinguishable from hybrid $\mathcal{H}_{j^*}$.*

Corollary 7.10. *If R is a ring as in Lemma 7.7, then hybrid $\mathcal{H}_{2N-j^*-1}$ is computationally indistinguishable from hybrid $\mathcal{H}_{2N-j^*}$.*

Corollary 7.11. *If R is a ring as in Lemma 7.7, then hybrid $\mathcal{H}_{2N-2}$ is computationally indistinguishable from hybrid $\mathcal{H}_{2N-1}$.*

Indistinguishability of Inner Hybrids. We now state and prove the following lemma (we again assume throughout that in all guard matrices and block-diagonal matrices, the first and second sub-matrices correspond to the subspaces with dimensions m_1 and m_2, respectively).

Lemma 7.12. *If R is the output of a ring-embedded homomorphic synthesizer, then for each $j \in [2, j^* - 1]$, hybrid $\mathcal{H}_{j-1}$ is computationally indistinguishable from hybrid $\mathcal{H}_j$.*

Proof. Our overall goal in this lemma is to replace the guard sub-matrices

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

with those of the form

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

for uniformly random $\mathbf{X}_1$ and $\mathbf{X}_2$. As in Lemma 7.8, we need to be able to make this swap and still simulate all of the other terms needed in the auxiliary baby subspace hiding assumption hybrid, but it is a little bit trickier in this case since we have "two-sided" terms.

We will again make the observation that if we randomize $\mathbf{R}_{j,2}$ (i.e. swap it for a random $\mathbf{X}_1$) then $\mathbf{L}_{j+1,1}$ appears distributed uniformly at random (and thus identically to $\mathbf{X}_2$), since it no longer has any relationships with any other guards. So without loss of generality, we may assume that we only need to randomize $\mathbf{R}_{j,2}$ rather than both $\mathbf{R}_{j,2}$ and $\mathbf{L}_{j+1,1}$. We will use this simplification for the rest of this proof.

Next, let's examine the terms related to $\mathbf{R}_{j,2}$ that we will need to switch. Note that, since we are currently in hybrid j, it is the case that we have already switched the matrix $\mathbf{L}_{j,1}$ to a uniformly random matrix $\mathbf{Z} \in R^{n \times m}$. This is very important for our proof to go through, although it will be used in one of the lemmas we apply in this proof rather than directly. We will need to simulate the following terms in the hybrid slot:

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

$$\mathbf{G}_j = \begin{bmatrix} | & | \\ \mathbf{Z} & \mathbf{L}_{j,2} \\ | & | \end{bmatrix} \cdot \begin{bmatrix} \mathbf{X} & \mathbf{0}_R \\ \mathbf{0}_R & \mathbf{0}_R \end{bmatrix} \cdot \begin{bmatrix} -- & \mathbf{R}_{j,1} & -- \\ -- & \mathbf{R}_{j,2} \text{ or } \mathbf{X}_1 & -- \end{bmatrix},$$
$$\mathbf{G}_{j+1} = \begin{bmatrix} | & | \\ \mathbf{L}_{j+1,1} & \mathbf{L}_{j+1,2} \\ | & | \end{bmatrix} \cdot \begin{bmatrix} \mathbf{X} & \mathbf{0}_R \\ \mathbf{0}_R & \mathbf{0}_R \end{bmatrix}$$

2. **Challenge-Relevant Elements**: The challenger provides the adversary $\mathcal{A}$ with several challenge-relevant elements of the following form:

$$\mathbf{H}_2 = \begin{bmatrix} | & | \\ \mathbf{Z} & \mathbf{L}_{j,2} \\ | & | \end{bmatrix} \cdot \begin{bmatrix} \mathbf{X} & \mathbf{0}_R \\ \mathbf{0}_R & \mathbf{X} \end{bmatrix} \cdot \begin{bmatrix} -- & \mathbf{R}_{j,1} & -- \\ -- & \mathbf{R}_{j,2} \text{ or } \mathbf{X}_1 & -- \end{bmatrix},$$
$$\mathbf{H}_3 = \begin{bmatrix} | & | \\ \mathbf{L}_{j+1,1} & \mathbf{L}_{j+1,2} \\ | & | \end{bmatrix} \cdot \begin{bmatrix} \mathbf{X} & \mathbf{0}_R \\ \mathbf{0}_R & \mathbf{X} \end{bmatrix}$$

It turns out that (a relabeling of) Lemma A.5 (which we state and prove formally in Appendix A) states the following: *Let m, n_1 and n_2 be integers such that $m \geq 6(n_1 + n_2)\log|R|$ where R is the output space of a ring-embedded homomorphic synthesizer. Let the matrices $\mathbf{R}_{j-1,1} \in R^{n_1 \times 2m}$, $\mathbf{L}_{j,2} \in R^{2m \times n_2}$, $\mathbf{R}_{j,1} \in R^{n_1 \times 2m}$, and $\mathbf{R}_{j,2} \in R^{n_2 \times 2m}$, $\mathbf{L}_{j+1,1} \in R^{2m \times n_1}$, and $\mathbf{L}_{j+1,2} \in R^{2m \times n_2}$ be random subject to the constraint that, for all relevant i:*

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

Suppose we sample sets of two matrices $\mathbf{X}_{1,i} \in R^{n_1 \times n_1}, \mathbf{X}_{2,i} \in R^{n_2 \times n_2}$ for $i \in [N]$ (for some $N = \text{poly}(\lambda)$; λ being the security parameter), and let $\mathbf{Z} \in R^{2m \times n_2}$ and $\mathbf{Z}' \in R^{n_1 \times 2m}$ be sampled uniformly at random as well. We claim the following: given the terms

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

the following sets of tuples are indistinguishable:

$$(\mathbf{L}_{j,1}\mathbf{X}_{1,i}\mathbf{R}_{j,1} + \mathbf{Z}\mathbf{X}_{2,i}\mathbf{R}_{j,2}) \quad \text{and} \quad \left(\mathbf{L}_{j,1}\mathbf{X}_{1,i}\mathbf{Z}' + \mathbf{Z}\mathbf{X}_{2,i}\mathbf{R}_{j,2}\right)$$

□

Note that the first tuple in the above equation is the form of $\mathbf{H}_2$ in the case of hybrid $\mathcal{H}_{j-1}$, and the second tuple is the form of $\mathbf{H}_2$ in the case of hybrid $\mathcal{H}_j$. Moreover, note that the lemma provides all of the necessary terms to simulate all of the other elements needed in the auxiliary baby subspace hiding assumption. So, by invoking Lemma A.5, we complete the proof of Lemma 7.12.

We also state the following corollaries, the proofs of which follow from essentially identical arguments to those in the proof of Lemma 7.12 (just like the case for Lemma 7.8 and are hence not detailed. We note that the first and second corollaries here technically rely on Corollary A.6, which is just the "transpose" of Lemma A.4.

Corollary 7.13. *If R is the output of a ring-embedded homomorphic synthesizer, then for each $j \in [j^* + 1, N - 1]$, hybrid $\mathcal{H}_{j-1}$ is computationally indistinguishable from hybrid $\mathcal{H}_j$.*

Corollary 7.14. *If R is the output of a ring-embedded homomorphic synthesizer (RHS), then for each $j \in [N + 1, 2N - j^* - 1]$, hybrid $\mathcal{H}_{j-1}$ is computationally indistinguishable from hybrid $\mathcal{H}_j$.*

Corollary 7.15. *If R is the output of a ring-embedded homomorphic synthesizer (RHS), then for each $j \in [2N - j^* + 1, 2N - 2]$, hybrid $\mathcal{H}_{j-1}$ is computationally indistinguishable from hybrid $\mathcal{H}_j$.*

Indistinguishability of $\mathcal{H}_{N-1}$ and $\mathcal{H}_N$. Finally, we state and prove the following lemma (we again assume throughout that in all guard matrices and block-diagonal matrices, the first and second sub-matrices correspond to the subspaces with dimensions m_1 and m_2, respectively).

Lemma 7.16. *If R is the output of a ring-embedded homomorphic synthesizer, then hybrid $\mathcal{H}_{N-1}$ is computationally indistinguishable from hybrid $\mathcal{H}_N$.*

Proof. Note that in both the hybrids $\mathcal{H}_{N-1}$ and $\mathcal{H}_N$, 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 $\mathcal{H}_{N-1}$ and $\mathcal{H}_N$ is that the challenge element to be provided to the adversary $\mathcal{A}$ is additionally randomized in the final hybrid $\mathcal{H}_N$.

Recall that, in $\mathcal{H}_{N-1}$ (and all the hybrids before it), the challenge elements have the form

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

but in hybrid $\mathcal{H}_N$, the challenge elements have the form

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

Next, observe the following:

- The guard sub-matrix $\mathbf{L}_{j^*,1}$ is randomized in hybrid $\mathcal{H}_{j^*-1}$ and in all hybrids through $\mathcal{H}_N$ (and in fact, more).
- The guard sub-matrix $\mathbf{R}_{j^*,1}$ is randomized in hybrid $\mathcal{H}_{N-1}$ and in all hybrids through $\mathcal{H}_N$.

In other words, in $\mathcal{H}_{N-1}$, 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. Let $\mathbf{T} \in \mathbb{R}^{2m \times 2m}$ be a uniformly random matrix. Now, by applying theorem 3.13 twice, we know that the tuples

$$(\mathbf{X}_{j^*,1}, \mathbf{Y}_{j^*,1}, \mathbf{X}_{j^*,1}\mathbf{X}\mathbf{Y}_{j^*,1}) \text{ and } (\mathbf{X}_{j^*,1}, \mathbf{Y}_{j^*,1}, \mathbf{T})$$

are indistinguishable from each other for random $\mathbf{X} \in R^{n \times n}$. This immediately implies that $\mathbf{C}_0$ and $\mathbf{C}_1$ are indistinguishable from each other, since the extra terms in $\mathbf{C}_1$ are independent of all of the terms in the above statement.

Note that we are given the only two guard matrices involved in this indistinguishability as part of the problem statement, so we can simulate all of the remaining guards (and thus terms) needed in the auxiliary baby subspace hiding assumption. So, any adversary that can distinguish between hybrids $\mathcal{H}_{N-1}$ and $\mathcal{H}_N$ breaks theorem 3.13, completing the proof of Lemma 7.16. □

Putting Everything Together. Finally, the proof of Lemma 7.7 follows by putting together all of the hybrid arguments we have shown above. This completes the proof of the auxiliary baby subspace hiding assumption as in Definition 7.6.

7.4 Proof of Baby Subspace Hiding Assumption-1

In this subsection, we prove that the baby subspace hiding assumption-1 as in Definition 5.3 (i,e., the baby version of the subspace hiding assumption-1 as in Definition 5.1) holds over any ring R as in Theorem 5.6. More formally, we prove the following.

Lemma 7.17. *Let R be any generic ring such that for any $m, Q = \text{poly}(\lambda)$ where m is any arbitrary positive integer satisfying $m > 3\lambda$ and Q is any arbitrary positive integer, we have*

$$(\mathbf{A}, \mathbf{A}\mathbf{K}) \overset{c}{\approx} (\mathbf{A}, \mathbf{U}), \quad (\mathbf{B}, \mathbf{K}'\mathbf{B}) \overset{c}{\approx} (\mathbf{B}, \mathbf{V})$$

where $\mathbf{K}, \mathbf{K}' \leftarrow R^{m \times m}$, $\mathbf{A}, \mathbf{U} \leftarrow R^{Qm \times m}$, and $\mathbf{B}, \mathbf{V} \leftarrow R^{m \times Qm}$. The baby subspace hiding assumption-1 as in Definition 5.3 holds over the ring R.

7.4.1 Overview of Proof Strategy

Once again, before we present a formal proof, we will provide additional intuition on why the assumption holds true and why the proof works. We will once again consider an instance of the assumption experiment with $z = (L+N) = 4$, and suppose that the adversary $\mathcal{A}$ chooses the challenge-index $j^* = 3$ (recall that for the baby subspace hiding assumption-1, we have $T = 2$ and $Z = 0$, and the challenge permutation slot-pair is $(1, 2)$ by default).

We will let $\mathbf{P}_{i,j}$ for $i \in [1, 4]$ and $j \in \{1, 2\}$ denote random 5×5 blockwise-permutation matrices over R. We emphasize that, for all i, $\mathbf{P}_{i,1}$ and $\mathbf{P}_{i,2}$ *represent the same permutation* and thus have the same blockwise structure, although the ring elements actually present in each matrix are uniformly random. In addition, recall that $\mathbf{R}_{i,j} \in R^{5m \times n}$ and $\mathbf{L}_{i,j} \in R^{n \times 5m}$ for $i \in [4]$ and $j \in \{1, 2\}$ denote our guard sub-matrices. Since these are really "blockwise" guards, we will label them $\mathbf{R}_{i,j,k}$ and $\mathbf{L}_{i,j,k}$ for $k \in [5]$ when it is useful to us. We need to do this when we need more fine-grained control over which terms in the permutation matrices the guards affect.

So, in our toy example, the challenger needs to provide $\mathcal{A}$ with terms of the following form:

1. **Generators**:

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

$$\mathbf{G}_1 = \begin{bmatrix} \mathbf{P}_{1,1} & \mathbf{0}_R \end{bmatrix} \cdot \begin{bmatrix} -- & \mathbf{R}_{1,1} & -- \\ -- & \mathbf{R}_{1,2} & -- \end{bmatrix},$$

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

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

$$\mathbf{G}_4 = \begin{bmatrix} | & | \\ \mathbf{L}_{4,1} & \mathbf{L}_{4,2} \\ | & | \end{bmatrix} \cdot \begin{bmatrix} \mathbf{P}_{4,1} \\ \mathbf{0}_R \end{bmatrix}$$

2. **Challenge-Relevant Elements.**

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

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

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

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

$$\mathbf{H}_4 = \begin{bmatrix} | & | \\ \mathbf{L}_{4,1} & \mathbf{L}_{3,2} \\ | & | \end{bmatrix} \cdot \begin{bmatrix} \mathbf{P}_{4,1} \\ \mathbf{P}_{4,2} \end{bmatrix}$$

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 $j^* = 3$):

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

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):

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

or

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

Intuitively, the security of this assumption follows from a similar principle as the auxiliary baby subspace hiding assumption in Appendix 5 (Definition 7.6). All elements *not* corresponding to the challenge-index (the generators and the challenge-relevant elements) are either random in the first *and* second slots, or random in the first slot and *all-zero* in the second slot. So all possible "legitimate" multiplications that can be performed efficiently using these elements 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). In this case, we will always output a valid permutation matrix.

Once again, 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.

Finally, note that since $\mathbf{P}_{i,1}$ and $\mathbf{P}_{i,2}$ represent identical permutations, we have

$$\prod_{i=1}^{4} \mathbf{P}_{i,1} = \prod_{i=1}^{4} \mathbf{P}_{i,2},$$

and the overall sum

$$\prod_{i=1}^{4} \mathbf{P}_{i,1} + \prod_{i=1}^{4} \mathbf{P}_{i,2}$$

is a blockwise permutation matrix representing the same permutation as both of the products.

Distribution Requirements. What do we *need* to leak in the terms provided by the adversary? There are two main constraints. The output should either be a blockwise permutation matrix equal to $\prod_{i=1}^{4} \mathbf{P}_{i,1} = \prod_{i=1}^{4} \mathbf{P}_{i,2}$ or it should be zero. However, given these constraints, we would obviously like the terms provided by the challenger to the adversary $\mathcal{A}$ to be "as random as possible."

Note that this is different from in the auxiliary baby subspace hiding assumption, where we only required that the output was zero or random. Here, we need the output matrices to be either zero or "blockwise permutation random."

The "zero" products are precisely as before in the auxiliary baby subspace hiding assumption: suppose we (informally) label the elements provided by the challenger to the adversary $\mathcal{A}$ in the auxiliary baby subspace hiding 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:

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

Preserving Permutation Outputs In the auxiliary baby subspace hiding assumption proof, we showed that we could completely randomize some of the guard matrices such that no adversary could tell the difference. In this case–for the baby subspace hiding assumption-1–this is clearly not possible: doing so will break the products of the permutations. However, we still need to be able to randomize some of the guard matrices, or else we will not be able to finish the proof. So what can we do?

Let's examine the first hybrid from the auxiliary baby subspace hiding assumption. We started wtih the guard matrices

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

and argued that they could be switched to

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

for a uniformly random $\mathbf{X}$. This clearly doesn't work in this case: the product

$$\begin{bmatrix} \mathbf{P}_{1,1} & \mathbf{P}_{1,2} \end{bmatrix} \cdot \begin{bmatrix} -- & -- & -- \\ -- & \mathbf{R}_{1,1} & -- \\ -- & \mathbf{R}_{1,2} & -- \end{bmatrix} \cdot \begin{bmatrix} | & | \\ \mathbf{L}_{2,1} & \mathbf{L}_{2,2} \\ | & | \end{bmatrix}$$

will be distributed as a blockwise permutation of the same form as both $\mathbf{P}_{1,1}$ and $\mathbf{P}_{1,2}$, while the product

$$\begin{bmatrix} \mathbf{P}_{1,1} & \mathbf{P}_{1,2} \end{bmatrix} \cdot \begin{bmatrix} -- & -- & -- \\ -- & \mathbf{X} & -- \\ -- & \mathbf{R}_{1,2} & -- \end{bmatrix} \cdot \begin{bmatrix} | & | \\ \mathbf{L}_{2,1} & \mathbf{L}_{2,2} \\ | & | \end{bmatrix}$$

will be a uniform-looking square matrix. So what can we do instead?

Recall that $\mathbf{R}_{1,2}$ has the following properties: $\mathbf{R}_{1,2} \cdot \mathbf{L}_{2,1} = \mathbf{0}_R$, and $\mathbf{R}_{1,2} \cdot \mathbf{L}_{2,2} = \mathbf{D}$ for some 5×5 blockwise diagonal matrix $\mathbf{D}$. How can we relax these properties enough to prove the security of our baby subspace hiding assumption-1 while still making sure our terms are consistent? Suppose we sample $\mathbf{X}$ such that $\mathbf{X} \cdot \mathbf{L}_{2,1}$ and $\mathbf{X} \cdot \mathbf{L}_{2,2}$ are *both* random 5×5 block diagonal matrices. This gives us an extra "small" block of randomness, as the following comparison shows. We have

$$\begin{bmatrix} -- & -- & \mathbf{R}_{1,2,1} & -- & -- \\ -- & -- & \mathbf{R}_{1,2,2} & -- & -- \\ -- & -- & \mathbf{R}_{1,2,3} & -- & -- \\ -- & -- & \mathbf{R}_{1,2,4} & -- & -- \\ -- & -- & \mathbf{R}_{1,2,5} & -- & -- \end{bmatrix} \begin{bmatrix} | & | & \dots & | & | \\ | & | & \dots & | & | \\ \mathbf{L}_{2,1,1} & \mathbf{L}_{2,1,2} & \dots & \mathbf{L}_{2,2,4} & \mathbf{L}_{2,2,5} \\ | & | & \dots & | & | \\ | & | & \dots & | & | \end{bmatrix}$$

$$= \begin{bmatrix} \mathbf{0}_R & & & \mathbf{Y}_{1,2} & & \\ & \mathbf{0}_R & & & \mathbf{Y}_{2,1} & \\ & & \mathbf{0}_R & & & \mathbf{Y}_{3,2} \\ \mathbf{0}_R & & & \mathbf{0}_R & & \\ & \mathbf{0}_R & & & \mathbf{Y}_{4,2} & \\ & & \mathbf{0}_R & & & \mathbf{Y}_{5,2} \end{bmatrix}$$

for random $\mathbf{Y}_{i,j}$ and non-written (i.e. off-diagonal) entries zero, and also

$$\begin{bmatrix} -- & -- & \mathbf{X}_1 & -- & -- \\ -- & -- & \mathbf{X}_2 & -- & -- \\ -- & -- & \mathbf{X}_3 & -- & -- \\ -- & -- & \mathbf{X}_4 & -- & -- \\ -- & -- & \mathbf{X}_5 & -- & -- \end{bmatrix} \begin{bmatrix} | & | & \dots & | & | \\ | & | & \dots & | & | \\ \mathbf{L}_{2,1,1} & \mathbf{L}_{2,1,2} & \dots & \mathbf{L}_{2,2,4} & \mathbf{L}_{2,2,5} \\ | & | & \dots & | & | \\ | & | & \dots & | & | \end{bmatrix}$$

$$= \begin{bmatrix} \mathbf{Y}_{1,1} & & & & \mathbf{Y}'_{1,2} & & & \\ & \mathbf{Y}_{2,1} & & & & \mathbf{Y}'_{2,1} & & \\ & & \mathbf{Y}_{3,1} & & & & \mathbf{Y}'_{3,2} & \\ & & & \mathbf{Y}_{4,1} & & & & \mathbf{Y}'_{4,2} \\ & & & & \mathbf{Y}_{5,1} & & & \mathbf{Y}'_{5,2} \end{bmatrix}$$

Slicing Down to the Auxiliary Baby Subspace Hiding Assumption. The above description looks fairly complicated. But what if we only consider certain rows and columns to simplify things? To start, let's just consider the top, left, and top-left components: in other words, $\mathbf{R}_{1,1,1}$, $\mathbf{R}_{1,2,1}$, $\mathbf{L}_{2,1,1}$, $\mathbf{L}_{2,2,2}$, and of course $\mathbf{X}_1$. We have the following relationships:

$$\begin{bmatrix} -- & \mathbf{R}_{1,1,1} & -- \\ -- & \mathbf{R}_{1,2,1} & -- \end{bmatrix} \cdot \begin{bmatrix} | & | \\ \mathbf{L}_{2,1,1} & \mathbf{L}_{2,2,1} \\ | & | \end{bmatrix} = \begin{bmatrix} \mathbf{Y}_{1,1} & \\ & \mathbf{Y}_{2,2} \end{bmatrix}$$

$$\begin{bmatrix} -- & \mathbf{R}_{1,1,1} & -- \\ -- & \mathbf{X}_1 & -- \end{bmatrix} \cdot \begin{bmatrix} | & | \\ \mathbf{L}_{2,1,1} & \mathbf{L}_{2,2,1} \\ | & | \end{bmatrix} = \begin{bmatrix} \mathbf{Y}_{1,1} & \\ \mathbf{Y}_{2,1} & \mathbf{Y}'_{2,2} \end{bmatrix}$$

In other words, $\mathbf{X}_1$ is completely independent of $\mathbf{R}_{1,1,1}$, $\mathbf{L}_{2,1,1}$, and $\mathbf{L}_{2,2,1}$. This should be a nice hint that we can reduce down to the auxiliary baby subspace hiding assumption.

For now, suppose we assume that all permutation matrices $\mathbf{P}_{i,j}$ are the identity. This won't change our proof much at this stage–it will only change which $\mathbf{R}_{i,j}$ and $\mathbf{L}_{i,j}$ terms are matched with each other at different levels of the construction, and allowing us to assume the permutations are the identity enables us to simplify a lot of the notation. If we only focus on the terms above, our challenge elements look like the following:

1. **Generators**:

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

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

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

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

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

2. **Challenge-Relevant Elements.**

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

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

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

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

$$\mathbf{H}_4 = \begin{bmatrix} | & | \\ \mathbf{L}_{4,1,1} & \mathbf{L}_{4,2,1} \\ | & | \end{bmatrix} \cdot \begin{bmatrix} \mathbf{X} \\ \mathbf{X} \end{bmatrix}$$

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 $j^* = 3$):

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

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):

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

or

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

What is this, exactly? An astute reader might notice that it is just the auxiliary baby subspace hiding assumption with extra indices. All of the matrices have the exact same properties and distributions as they would in the auxiliary baby subspace hiding assumption. One again, as before, we can completely randomize the blue columns and rows (relative to the rows and columns presented here–*not* other terms we aren't considering here) and the distribution will be unchanged.

Note that we still need to simulate all of the other (orthogonal to these) rows and columns for the full assumptions and proofs, but we can do that using our generalized "lifting" lemma, which is Lemma 7.2 (Appendix 7.1).

Proof Strategy. Our proof strategy for the baby subspace hiding assumption-1 can be viewed as a sort of application of the proof and proof techniques of the auxiliary baby subspace hiding assumption. We essentially will use 6 meta-hybrids, and astute reader can probably guess what they will be. Suppose the challenge term is in index j. Our hybrids look like the following:

- $\mathcal{MH}_0$: the real scheme with the challenge term $\mathbf{C}_0$ used.
- $\mathcal{MH}_1$: same as $\mathcal{MH}_0$, except for the following: we use a challenge term $\mathbf{C}_1$ such that instead of using the full permutation matrix $\mathbf{P}_{j,2}$, we use a matrix that is uniformly zero *except* in the block of the first row in the same slot where $\mathbf{P}_{j,2}$ is not zero.
- $\mathcal{MH}_2$: same as $\mathcal{MH}_1$, except now we add another nonzero block. we use a challenge term $\mathbf{C}_1$ such that instead of using the full permutation matrix $\mathbf{P}_{j,2}$, we use a matrix that is uniformly zero *except* in the blocks in the first *two rows* in the same slots where $\mathbf{P}_{j,2}$ is not zero.
- $\mathcal{MH}_3$: same as above, but we add the block from the third row in $\mathbf{P}_{j,2}$.
- $\mathcal{MH}_4$: same, but we add the block from the fourth row in $\mathbf{P}_{j,2}$.
- $\mathcal{MH}_5$: we use all of the blocks from $\mathbf{P}_{j,2}$, meaning we are in the real scheme with the challenge term $\mathbf{C}_1$ used.

Note that moving between each one of these hybrids essentially boils down to an application of the auxiliary baby subspace hiding assumption where we simulate all of the rows/columns not directly "touching" the block of the permutation matrix we are switching. In other words, if all of the permutation matrices are blockwise-identity, the first meta-hybrid will modify the submatrices of the form $\mathbf{R}_{i,j,1}$ and $\mathbf{L}_{i,j,1}$ and the first row of the "challenge permutation," the second meta-hybrid will modify the submatrices of the form $\mathbf{R}_{i,j,2}$ and $\mathbf{L}_{i,j2}$ and the second row of the "challenge permutation," and so forth. Non-identity permutations will mean that indices will be mixed and matched, but the proof idea is exactly the same (just the notation is considerably more difficult).

With this in mind, we can continue on to the formal version of the proof.

7.4.2 Formal Proof of Baby Subspace Hiding Assumption-1.

We now formally prove the baby subspace hiding assumption-1 as in Definition 5.3, i.e., we formally prove Lemma 7.17.

More precisely, we will defined a series of hybrids $\mathcal{MH}_i$ for $i \in [0,5]$ such that any adversary that can distinguish between hybrids $\mathcal{MH}_i$ and $\mathcal{MH}_{i+1}$ can be used to break the auxiliary baby subspace hiding assumption (as defined in Definition 7.6).

We formally define our hybrids in Figure 16. Note that the only difference between our hybrid description and the baby subspace hiding assumption-1 occurs in the description of the challenge element, where we iteratively add blocks to the challenge term.

We can now formally prove security.

Lemma 7.18. *Any adversary $\mathcal{A}$ that can distinguish experiment $\mathsf{Expt}^{\mathcal{MH}_j}_{R,m,L,N,T}$: from experiment $\mathsf{Expt}^{\mathcal{MH}_{j+1}}_{R,m,L,N,T}$ for $j \in [0,4]$ can be used to break the (auxiliary) baby subspace hiding assumption as defined in 7.5 for parameters $N' = (L+N)$ and $m_1 = m_2 = m$.*

Proof. Suppose we are given an instance of the auxiliary baby subspace hiding assumption. We must correctly simulate experiment $\mathsf{Expt}^{\mathcal{MH}_j}_{R,m,L,N,T}$ if we are given the challenge element $\mathbf{C}_0$ and experiment $\mathsf{Expt}^{\mathcal{MH}_{j+1}}_{R,m,L,N,T}$ if we are given the challenge element $\mathbf{C}_1$.

Relabeling the Guards. The first thing we will do is relabel the guard matrices. Note that the final product of all (legal) combinations of terms can be written as a sum of (at most) 10 products, each of which is of the form (permutation * guard * permutation * guard * .... * guard * permutation). This is simple to see if the permutation matrices are all the identity: any (legal) final product has no terms that contain products between guard matrices with different indices.

For sets of arbitrary permutation matrices, each legal final product can be written as a sum of products such that each product contains a unique set of guard matrices. However, unlike in the case where all permutations are the identity matrix, these "sets" of guards may not all have the same index. To help clarify our proof, we will relabel our guards based on which "product" they appear in our challenge terms.

For $k \in [5]$ and for all $i \in [L+N-1]$, let $\widetilde{\mathbf{R}_{i,1,k}}$, $\widetilde{\mathbf{R}_{i,2,k}}$, $\widetilde{\mathbf{L}_{i+1,1,k}}$, $\widetilde{\mathbf{L}_{i+1,2,k}}$ denote the guard matrices that, in a legal multiplication of terms, are (potentially) included with the kth block of their corresponding block diagonal matrix *in the challenge term*. Note that this relabeling does not change the distribution of the guards at all: it just lets us keep track of which different (and uncorrelated) sets of guards appear in products together.

Using the Auxiliary baby subspace hiding assumption. Note that if we set all of the guards not of the form $\widetilde{\mathbf{R}_{i,1,j}}$, $\widetilde{\mathbf{R}_{i,2,j}}$, $\widetilde{\mathbf{L}_{i+1,1,j}}$, and $\widetilde{\mathbf{L}_{i+1,2,j}}$ for all $i \in [L+N]$ in our hybrids equal to zero, we will be left with exactly an instance of the auxiliary baby subspace hiding assumption. This is what we "drew" in the intuition section above. However, performing this action would lead us to a proof in the wrong direction.

So, our proof will proceed as follows: given an instance of the auxiliary baby subspace hiding assumption, we will "map" it to levels and guards corresponding to $\widetilde{\mathbf{R}_{i,1,j}}$, $\widetilde{\mathbf{R}_{i,2,j}}$, $\widetilde{\mathbf{L}_{i+1,1,j}}$, and $\widetilde{\mathbf{L}_{i+1,2,j}}$ for all $i \in [L+N]$. We can then apply our "lifting" lemma (Lemma 7.2) (since all of our terms have the appropriate form) to generate all of the rest of the guard matrices, which are orthogonal to the ones we have derived from applying Lemma 7.2 to our baby subspace hiding assumption terms. This allows us to simulate all of the terms we need in the experiments.

Experiment $\mathsf{Expt}^{\mathcal{MH}_j}_{R,m,L,N,T}$:

1. The adversary $\mathcal{A}$ chooses a challenge-index $i^* \in [N]$, and a program $\{P\}$ where $P = \{N_{\ell,b}\}_{\ell\in[L],b\in\{0,1\}}$. It provides $(i^*, \{P\})$ to the challenger.
2. The challenger provides the adversary with the following matrices sampled from appropriate distributions (in what follows, all permutation block diagonal matrices $\mathbf{D}$ are of dimension $10m \times 10m$, and a total of $(L + N)$ guard matrix pairs are used):

   (a) **Generators:**

   - For each $\ell \in [L]$, $b \in \{0, 1\}$: many samples of the form $\mathbf{L}_\ell \mathbf{D}^{(1,N_{\ell,b})} \mathbf{R}_\ell$, where $\mathbf{D}^{(1,(N_{\ell,b})}$ is a permutation-embedded block-diagonal matrix sampled from the appropriate distribution.
   - For each $i \in [N]$: many samples of the form $\mathbf{L}_{L+i} \mathbf{D}^{(1,I)} \mathbf{R}_{L+i}$, where $I \in \{0, 1\}^{5\times5}$ denotes the identity permutation, and $\mathbf{D}^{(1,I)}$ is an identity-permutation-embedded block-diagonal matrix sampled from the appropriate distribution.

   In each case, the "end guards" $\mathbf{L}_1$ and $\mathbf{R}_{L+N}$ are specially structured, as described in Section 4.3.

   (b) **Challenge-Relevant Elements**:

   - For each $\ell \in [L]$ and for each $b \in \{0, 1\}$: many samples of the form $\mathbf{L}_\ell \mathbf{D}^{(1,N_{\ell,b}),(2,N_{\ell,b})} \mathbf{R}_\ell$, where $\mathbf{D}^{(1,N_{\ell,b}),(2,N_{\ell,b})}$ is a permutation-embedded block-diagonal matrix sampled from the appropriate distribution.
   - For each $i \in [N]$: many samples of the form $\mathbf{L}_{L+i} \mathbf{D}^{(1,I),(2,I)} \mathbf{R}_{L+i}$, where $I \in \{0, 1\}^{5\times5}$ denotes the identity permutation, and $\mathbf{D}^{(1,I),(2,I)}$ is an identity-permutation-embedded block-diagonal matrix sampled from the appropriate distribution.

   Once again, in each case, the "end guards" $\mathbf{L}_1$ and $\mathbf{R}_{L+N}$ are specially structured, as described in Section 4.3.

   (c) **Special Challenge-Index Elements**: Many samples of the form $\mathbf{L}_{(L+i^*)} \mathbf{D}^{2,I)} \mathbf{R}_{(L+i^*)}$, where $I \in \{0, 1\}^{5\times5}$ again denotes the identity permutation, and $\mathbf{D}^{(2,I)}$ is an identity-permutation-embedded block-diagonal matrix sampled from the appropriate distribution.

   Also, in the case when $i^* = N$, the right "end guard" $\mathbf{R}_{L+N}$ is specially structured, as described in Section 4.3.

   (d) **The Challenge Element**: A sample that has one the following form. Let $\mathbf{E}_j \in \{0, 1\}^{5\times m}$ be a diagonal matrix with the first j entries one and the last $5 - j$ blocks zero. The challenge element is as follows:

   - $\mathbf{L}_{(L+i^*)} \mathbf{D}^{(1,I),(2,\mathbf{E}_j)} \mathbf{R}_{(L+i^*)}$,

   where $I \in \{0, 1\}^{5\times5}$ again denotes the identity permutation and $\mathbf{D}^{(1,I),(2,\mathbf{E}_j)}$ is a block-diagonal matrix sampled from the appropriate distribution.

   Also, in the case when $i^* = N$, the right "end guard" $\mathbf{R}_{L+N}$ is specially structured, as described in Section 4.3.

Figure 16: Hybrid Experiments for the proof of Baby Subspace Hiding Assumption-1

Furthermore, note that if we were given $\mathbf{C}_0$ as a challenge term as part of the auxiliary baby subspace hiding assumption description, we have successfully simulated $\mathsf{Expt}^{\mathcal{MH}_j}_{R,m,L,N,T}$, and if we were given $\mathbf{C}_1$, we have successfully simulated $\mathsf{Expt}^{\mathcal{MH}_{j+1}}_{R,m,L,N,T}$, meaning our simulation is correct, which completes the proof.

7.5 Proof of Baby Subspace Hiding Assumption-2

In this section, we prove that the baby subspace hiding assumption-2 as in Definition 5.4 (i,e., the baby version of the subspace hiding assumption-2 as in Definition 5.2) holds over any ring R as in Theorem 5.6. In particular, we prove the following lemma:

Lemma 7.19. *Let R be any generic ring such that for any $m, Q = \mathrm{poly}(\lambda)$ where m is any arbitrary positive integer satisfying $m > 3\lambda$ and Q is any arbitrary positive integer, we have*

$$(\mathbf{A}, \mathbf{AK}) \overset{c}{\approx} (\mathbf{A}, \mathbf{U}), \quad (\mathbf{B}, \mathbf{K}'\mathbf{B}) \overset{c}{\approx} (\mathbf{B}, \mathbf{V})$$

where $\mathbf{K}, \mathbf{K}' \leftarrow R^{m\times m}$, $\mathbf{A}, \mathbf{U} \leftarrow R^{Qm\times m}$, and $\mathbf{B}, \mathbf{V} \leftarrow R^{m\times Qm}$. The baby subspace hiding assumption-2 as defined in Definition 5.4 holds over the ring R.

This is a proof that is very similar to the full permutation subspace hiding assumption, and there are only a couple of extra steps we need.

7.5.1 Overview of Proof Strategy

Let's again start with a graphical overview. We will once again consider an instance of the assumption experiment with $z = (L + N) = 4$, and suppose that the adversary $\mathcal{A}$ chooses the challenge-index $j^* = 3$ (recall that for the baby subspace hiding assumption-2, we have $T = 1$ and $Z = 1$, and the challenge slot-pair is $(1, 2)$ by default).

We will let $\mathbf{P}_i$ for $i \in [1, 4]$ denote random 5×5 blockwise-permutation matrices over R. Unlike in the permutation subspace hiding assumption, here we will also have fully random matrices: we will let $\mathbf{Z}_i$ for $i \in [4]$ denote fully uniform random matrices. In addition, recall that $\mathbf{R}_{i,j}$ and $\mathbf{L}_{i,j}$ for $i \in [4]$ and $j \in [2]$ denote our guard matrices. Since these are really "blockwise" guards, we will once again label them $\mathbf{R}_{i,j,k}$ and $\mathbf{L}_{i,j,k}$ for $k \in [5]$ when it is useful to us. We need to do this when we need more fine-grained control over which terms in the permutation matrices the guards affect.

So, in our toy example, the challenger needs to provide $\mathcal{A}$ with terms of the following form:

1. **Generators**:

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

$$\mathbf{G}_1 = \begin{bmatrix}\mathbf{Z}_1 & \mathbf{0}_R\end{bmatrix} \cdot \begin{bmatrix}-- & \mathbf{R}_{1,1} & -- \\ -- & \mathbf{R}_{1,2} & --\end{bmatrix},$$

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

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

$$\mathbf{G}_4 = \begin{bmatrix}| & | \\ \mathbf{L}_{4,1} & \mathbf{L}_{4,2} \\ | & |\end{bmatrix} \cdot \begin{bmatrix}\mathbf{Z}_4 \\ \mathbf{0}_R\end{bmatrix}$$

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):

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

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

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

$$\mathbf{H}_4 = \begin{bmatrix}| & | \\ \mathbf{L}_{4,1} & \mathbf{L}_{3,2} \\ | & |\end{bmatrix} \cdot \begin{bmatrix}\mathbf{Z}_4 \\ \mathbf{P}_4\end{bmatrix}$$

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 $j^* = 3$):

$$\mathbf{S} = \begin{bmatrix}| & | \\ \mathbf{L}_{3,1} & \mathbf{L}_{3,2} \\ | & |\end{bmatrix} \cdot \begin{bmatrix}\mathbf{0}_R & \mathbf{0}_R \\ \mathbf{0}_R & \mathbf{P}_3\end{bmatrix} \cdot \begin{bmatrix}-- & \mathbf{R}_{3,1} & -- \\ -- & \mathbf{R}_{3,2} & --\end{bmatrix}$$

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):

$$\mathbf{C}_0 = \begin{bmatrix}| & | \\ \mathbf{L}_{3,1} & \mathbf{L}_{3,2} \\ | & |\end{bmatrix} \cdot \begin{bmatrix}\mathbf{Z}_3 & \mathbf{0}_R \\ \mathbf{0}_R & \mathbf{0}_R\end{bmatrix} \cdot \begin{bmatrix}-- & \mathbf{R}_{3,1} & -- \\ -- & \mathbf{R}_{3,2} & --\end{bmatrix}$$

or

$$\mathbf{C}_1 = \begin{bmatrix}| & | \\ \mathbf{L}_{3,1} & \mathbf{L}_{3,2} \\ | & |\end{bmatrix} \cdot \begin{bmatrix}\mathbf{Z}_3 & \mathbf{0}_R \\ \mathbf{0}_R & \mathbf{P}_3\end{bmatrix} \cdot \begin{bmatrix}-- & \mathbf{R}_{3,1} & -- \\ -- & \mathbf{R}_{3,2} & --\end{bmatrix}$$

This set of terms initially seems slightly more complicated than before in the regular permutation baby subspace hiding assumption: now we have to deal with both uniformly random matrices and permutation matrices, and keep track of them both. However, this is slightly deceptive: "proper" multiplications of sets of four terms will either result in a uniformly random matrix or a uniformly zero matrix. There is no way for an adversary to properly multiply and extract anything permutation-related (in fact, this is the intuition for why we can "eliminate" permutations).

From a mechanical perspective, our proof is more complicated than that of the permutation baby subspace hiding assumption. Suppose we again try to "slice down" the components of the permutation matrices into a baby subspace hiding instance (each with the larger-dimensional uniform block). We will run into a slight difficulty: the (auxiliary) baby subspace hiding assumption requires that guard matrices multiply with their corresponding guard matrices to a

random block, while the larger-dimensional block with uniform matrices multiplies sets of 5 guard matrices–so the overall product is a block diagonal.

Graphically speaking, our "real" sliced-down distribution of guards would look like the following:

$$\begin{bmatrix} -- & -- & \mathbf{R}_{1,1,1} & -- & -- \\ -- & -- & \mathbf{R}_{1,2,1} & -- & -- \\ -- & -- & \mathbf{R}_{1,2,2} & -- & -- \\ -- & -- & \mathbf{R}_{1,2,3} & -- & -- \\ -- & -- & \mathbf{R}_{1,2,4} & -- & -- \\ -- & -- & \mathbf{R}_{1,2,5} & -- & -- \end{bmatrix} \begin{bmatrix} | & | & | & | & | & | \\ | & | & | & | & | & | \\ \mathbf{L}_{2,1,1} & \mathbf{L}_{2,2,1} & \mathbf{L}_{2,2,2} & \mathbf{L}_{2,2,3} & \mathbf{L}_{2,2,4} & \mathbf{L}_{2,2,5} \\ | & | & | & | & | & | \\ | & | & | & | & | & | \end{bmatrix}$$

$$= \begin{bmatrix} \mathbf{Y}_1 & & & & & \\ & \mathbf{Y}_2 & & & & \\ & & \mathbf{Y}_3 & & & \\ & & & \mathbf{Y}_4 & & \\ & & & & \mathbf{Y}_5 & \\ & & & & & \mathbf{Y}_6 \end{bmatrix}$$

for random $\mathbf{Y}_i$ and non-written (i.e. off-diagonal) entries zero. However, to apply the baby subspace assumption (with non-equal dimensions), we would need the product of the guards to look like the following:

$$\begin{bmatrix} -- & -- & \mathbf{R}_{1,1,1} & -- & -- \\ -- & -- & \mathbf{R}_{1,2,1} & -- & -- \\ -- & -- & \mathbf{R}_{1,2,2} & -- & -- \\ -- & -- & \mathbf{R}_{1,2,3} & -- & -- \\ -- & -- & \mathbf{R}_{1,2,4} & -- & -- \\ -- & -- & \mathbf{R}_{1,2,5} & -- & -- \end{bmatrix} \begin{bmatrix} | & | & | & | & | & | \\ | & | & | & | & | & | \\ \mathbf{L}_{2,1,1} & \mathbf{L}_{2,2,1} & \mathbf{L}_{2,2,2} & \mathbf{L}_{2,2,3} & \mathbf{L}_{2,2,4} & \mathbf{L}_{2,2,5} \\ | & | & | & | & | & | \\ | & | & | & | & | & | \end{bmatrix}$$

$$= \begin{bmatrix} \mathbf{Y} & \mathbf{Y} & \mathbf{Y} & \mathbf{Y} & \mathbf{Y} & \mathbf{0}_R \\ \mathbf{Y} & \mathbf{Y} & \mathbf{Y} & \mathbf{Y} & \mathbf{Y} & \mathbf{0}_R \\ \mathbf{Y} & \mathbf{Y} & \mathbf{Y} & \mathbf{Y} & \mathbf{Y} & \mathbf{0}_R \\ \mathbf{Y} & \mathbf{Y} & \mathbf{Y} & \mathbf{Y} & \mathbf{Y} & \mathbf{0}_R \\ \mathbf{Y} & \mathbf{Y} & \mathbf{Y} & \mathbf{Y} & \mathbf{Y} & \mathbf{0}_R \\ \mathbf{0}_R & \mathbf{0}_R & \mathbf{0}_R & \mathbf{0}_R & \mathbf{0}_R & \mathbf{Y}_1 \end{bmatrix}$$

where each $\mathbf{Y}$ is independent and random (i.e. not the same–we omit indices for ease of description). However, this is not too difficult to get around. In Lemma A.4, we show the following: suppose we have the following guard 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 show that, given $\mathbf{R}_1$, $\mathbf{L}_2$ and $\mathbf{R}_2$, $\mathbf{L}_1$, the following is indistinguishable from random for random $\mathbf{X}_1$, $\mathbf{X}_2$:

$$\begin{bmatrix} \mathbf{X}_1 & \mathbf{X}_2 \end{bmatrix} \begin{bmatrix} -- & \mathbf{R}_1 & -- \\ -- & \mathbf{R}_2 & -- \end{bmatrix}$$

Since all of our guards corresponding to level 1 in our baby half permutation assumption only show up in a form where they are multiplied by random matrices and added together as in the above lemma statement, we can replace them with a random "mega-guard" in our assumption and the adversary cannot tell. We can then use our lifting lemma (Lemma 7.2) to simulate the rest of the guards and argue full indistinguishability.

Proof Strategy. Our overall proof hybrids are similar to those for the (regular) permutation baby subspace hiding assumption, which we use again here. We again will use 6 meta-hybrids. Suppose the challenge term is in index j. Our hybrids look like the following:

- $\mathcal{MH}_0$: the real scheme with the challenge term $\mathbf{C}_0$ used.
- $\mathcal{MH}_1$: same as $\mathcal{MH}_0$, except for the following: we use a challenge term $\mathbf{C}_1$ such that instead of using the full permutation matrix $\mathbf{P}_{j,2}$, we use a matrix that is uniformly zero *except* in the block of the first row in the same slot where $\mathbf{P}_{j,2}$ is not zero.
- $\mathcal{MH}_2$: same as $\mathcal{MH}_1$, except now we add another nonzero block. we use a challenge term $\mathbf{C}_1$ such that instead of using the full permutation matrix $\mathbf{P}_{j,2}$, we use a matrix that is uniformly zero *except* in the blocks in the first *two rows* in the same slots where $\mathbf{P}_{j,2}$ is not zero.
- $\mathcal{MH}_3$: same as above, but we add the block from the third row in $\mathbf{P}_{j,2}$.
- $\mathcal{MH}_4$: same, but we add the block from the fourth row in $\mathbf{P}_{j,2}$.
- $\mathcal{MH}_5$: we use all of the blocks from $\mathbf{P}_{j,2}$, meaning we are in the real scheme with the challenge term $\mathbf{C}_1$ used.

However, proving indistinguishability of these hybrids is a little more complicated than before. Instead of just taking an instance of the baby subspace assumption, "lifting" it, and then simulating the rest of the terms as before, we need to first convert the guards in the first index to the appropriate form. So we will have $L + N$ hybrid arguments where we switch the guards in each slot to the baby subspace-amenable representation, an argument similar to that of the (regular) baby subspace hiding assumption where we "lift" a baby subspace assumption and simulate the terms, and then $L + N$ more hybrid arguments where we switch the guards back to the textbook form. This is a bit tedious, but it is substantially less tedious than proving a new, general form of the (auxiliary) baby subspace assumption itself.

7.5.2 Formal Proof of Baby Subspace Hiding Assumption-2

We now present a formal proof of the permutation baby subspace hiding assumption as defined in Definition 5.2, i.e., we formally prove Lemma 7.19.

More precisely, we will define a series of hybrids $\mathcal{MH}_i$ for $i \in [0, 5]$ such that any adversary that can distinguish between hybrids $\mathcal{MH}_i$ and $\mathcal{MH}_{i+1}$ can be used to either break the auxiliary baby subspace assumption (as in Definition 7.5) or Lemma A.4.

We will next formally argue that the above hybrids are indistinguishable.

Lemma 7.20. *Any adversary $\mathcal{A}$ that can distinguish experiment $\mathsf{Expt}^{\mathcal{MH}_j}_{R,m,L,N}$: from experiment $\mathsf{Expt}^{\mathcal{MH}_{j+1}}_{R,m,L,N}$ for $j \in [0, 4]$ can be used to break the (auxiliary) baby subspace assumption as defined in 7.5 or Lemma A.4.*

Proof. In order to complete this proof, we will define a set of hybrids $\mathcal{H}_i$ for $i \in [0, 2(L + N) - 1]$. In $\mathcal{H}_1$, we will simulate $\mathcal{MH}_j$, and in $\mathcal{H}_{2(L+N)-1}$, we will simulate $\mathcal{MH}_{j+1}$. Our hybrids are defined as follows:

1. $\mathcal{H}_0$: The experiment $\mathsf{Expt}^{\mathcal{MH}_j}_{R,m,L,N,b}$.
2. $\mathcal{H}_i$ for $i \in [L + N - 1]$: same as in $\mathcal{H}_i$, except we change the structure of the guard matrices $\mathbf{R}_i$ and $\mathbf{L}_{i+1}$. In particular, we replace the guard matrices $\mathbf{R}_{i,2,1}, ..., \mathbf{R}_{i,2,5}$ and $\mathbf{L}_{i+1,2,1}, ...\mathbf{L}_{i+1,2,5}$ with a single set of larger guard matrices $\mathbf{R}'_i$ and $\mathbf{L}'_{i+1}$, which are five times as tall and wide as an original guard, respectively.
3. $\mathcal{H}_{L+N}$: We switch the form of the challenge element from that of $\mathsf{Expt}^{\mathcal{MH}_j}_{R,m,L,N,b}$ to that of $\mathsf{Expt}^{\mathcal{MH}_j}_{R,m,L,N,b}$.
4. $\mathcal{H}_i$ for $i \in [2(L + N) - 1]$: we "undo" the guard matrix changes we made earlier in the hybrids. In particular, we replace our single set of larger guard matrices $\mathbf{R}'$ and $\mathbf{L}'$, which are five times as tall and wide as an original guard, respectively, with the original guard matrices $\mathbf{R}_{i,2,1}, ..., \mathbf{R}_{i,2,5}$ and $\mathbf{L}_{i+1,2,1}, ...\mathbf{L}_{i+1,2,5}$.

In lemmas 7.21 and 7.22, we show that each successive hybrid is indistinguishable, completing the proof. □

Lemma 7.21. *For $i \in [0, L + N - 2]$ and $i \in [L + N, 2(L + N) - 1]$, any adversary that can distinguish $\mathcal{H}_i$ from $\mathcal{H}_{i+1}$ can be used to break Lemma A.4.*

Experiment $\mathsf{Expt}^{\mathcal{MH}_j}_{R,m,L,N,b}$:

1. The adversary $\mathcal{A}$ chooses a challenge-index $\ell^* \in [L]$, a challenge-bit $\beta \in \{0,1\}$ and a program $\{P\}$, where $P = \{N_{\ell,b}\}_{\ell\in[L],b\in\{0,1\}}$. It provides $(\ell^*, \beta, \{P\})$ to the challenger.
2. The challenger provides the adversary with the following matrices sampled from appropriate distributions (in what follows, all block diagonal matrices $\mathbf{D}$ are of dimension $5m(T+Z) \times 5m(T+Z)$, and a total of $(L+N)$ guard matrix pairs are used):
   (a) **Generators:**
      - For each $\ell \in [L+N]$ and each $z \in [Z]$: many samples of the form $\mathbf{L}_\ell \mathbf{D}^{(1)} \mathbf{R}_\ell$.

   In each case, the "end guards" $\mathbf{L}_1$ and $\mathbf{R}_{L+N}$ are specially structured, as described in Section 4.3.

   (b) **Challenge-Relevant Elements**:
      - For each $\ell \in [L]$ and $b \in \{0,1\}$: many samples of the form $\mathbf{L}_\ell \mathbf{D}^{(1),(2,N_{\ell,b})} \mathbf{R}_\ell$, where $\mathbf{D}^{(1),(2,N_{\ell,b})}$ is a permutation-embedded block-diagonal matrix sampled from the appropriate distribution, with an R-embedding of the permutation matrix $N_{\ell,b}$ in the second diagonal block and a uniform sub-matrix in the first diagonal block.
      - For each $i \in [N]$: many samples of the form $\mathbf{L}_{L+i} \mathbf{D}^{(1),(2,I)} \mathbf{R}_{L+i}$, where $I \in \{0,1\}^{5\times 5}$ denotes the identity permutation and $\mathbf{D}^{(1),(2,I)}$ is a block-diagonal matrix sampled from the appropriate distribution, with an R-embedding of I in the second diagonal block and a uniform sub-matrix in the first diagonal block.

   Once again, in each case, the "end guards" $\mathbf{L}_1$ and $\mathbf{R}_{L+N}$ are specially structured, as described in Section 4.3.

   (c) **Special Challenge-Index Elements**: Many samples of the form $\mathbf{L}_{\ell^*} \mathbf{D}^{(2,N_{\ell^*,\beta})} \mathbf{R}_{\ell^*}$, where $\mathbf{D}^{(2,N^*,\ell^*,\beta)}$ is a permutation-embedded block-diagonal matrix sampled from the appropriate distribution.
   Also, in the case when $\ell^* = 1$, the left "end guard" $\mathbf{L}_1$ is specially structured, as described in Section 4.3.

   (d) **The Challenge Element**: Let $N^j_{\ell^*,\beta}$ denote the matrix $N_{\ell^*,\beta}$ with the last $5-j$ rows zeroed. The challenge element has the following form:
      - $\mathbf{L}_{\ell^*} \mathbf{D}^{(1),(2,N^j_{\ell^*,\beta})} \mathbf{R}_{\ell^*}$

   where:
      - $\mathbf{D}^{(1),(2,N^j_{\ell,\beta})}$ is a permutation-embedded block-diagonal matrix sampled from the appropriate distribution, with an R-embedding of the cropped permutation matrix $N^j_{\ell^*,\beta}$ in the second diagonal block and a uniform sub-matrix in the first diagonal block.

   Also, in the case when $\ell^* = 1$, the left "end guard" $\mathbf{L}_1$ is specially structured, as described in Section 4.3.

Figure 17: Hybrid Experiments for the proof of Baby Subspace Hiding Assumption-2

Proof. This proof is very similar to the proof of Lemma 7.8, and follows immediately from Lemma A.4 modulo some simple re-labeling. Hence, this proof is not detailed. □

Lemma 7.22. *Any adversary that can be used to distinguish $\mathcal{H}_{L+N-1}$ from $\mathcal{H}_{L+N}$ can be used to break the (auxiliary) baby subspace assumption as defined in 7.5.*

Proof. Suppose we are given an instance of the baby subspace assumption. We must correctly simulate experiment $\mathsf{Expt}^{\mathcal{MH}_j}_{R,m,L,N,T}$ (with the "corrupted" guard matrices if we are given the challenge element $\mathbf{C}_0$ and experiment $\mathsf{Expt}^{\mathcal{MH}_{j+1}}_{R,m,L,N,T}$ (also with the "corrupted" guard matrices) if we are given the challenge element $\mathbf{C}_1$.

Relabeling the Guards. The first thing we will do is relabel the guard matrices. Note that the final product of all (legal) combinations of terms can be written as a sum of (at most) 10 products, each of which is of the form (permutation * guard * permutation * guard * .... * guard * permutation). This is simple to see if the permutation matrices are all the identity: any (legal) final product has no terms that contain products between guard matrices with different indices.

For sets of arbitrary permutation matrices, each legal final product can be written as a sum of products such that each product contains a unique set of guard matrices. However, unlike in the case where all permutations are the identity matrix, these "sets" of guards may not all have the same index. To help clarify our proof, we will relabel our guards based on which "product" they appear in our challenge terms.

For $k \in [5]$ and for all $i \in [L+N-1]$, let $\widetilde{\mathbf{R}_{i,1,k}}$ and $\widetilde{\mathbf{L}_{i+1,1,k}}$ denote the guard matrices that, in a legal multiplication of terms, are (potentially) included with the kth block of the block diagonal permutation matrix *in the challenge term*. Note that this relabeling does not change the distribution of the guards at all: it just lets us keep track of which different (and uncorrelated) sets of guards appear in products together.

Using the Auxiliary Baby Subspace Hiding Assumption. Note that if we set all of the guards not of the form $\widetilde{\mathbf{R}_{i,j}}$, $\mathbf{R}'_i$, $\widetilde{\mathbf{L}_{i+1,j}}$, and $\mathbf{L}'_{i+1}$ for all $i \in [L+N]$ in our hybrids equal to zero, we will be left with exactly an instance of the baby subspace assumption. This is what we "drew" in the intuition section above. However, performing this action would lead us to a proof in the wrong direction.

So, our proof will proceed as follows: given an instance of the baby subspace assumption with parameters $N' = (L+N)$ and $(m_1, m_2) = (5m, m)$, we will "map" it to levels and guards corresponding to $\widetilde{\mathbf{R}_{i,j}}$, $\mathbf{R}'_i$, $\widetilde{\mathbf{L}_{i+1,j}}$, and $\mathbf{L}'_{i+1}$ for all $i \in [L+N]$ We can then apply our generalized "lifting" lemma (Lemma 7.2, Section 7.1) (since all of our terms have the appropriate form) to generate all of the rest of the guard matrices, which are orthogonal to the ones we have derived from applying Lemma 7.2 to our baby subspace assumption terms. This allows us to simulate all of the terms we need in the experiments.

Furthermore, note that if we were given $\mathbf{C}_0$ as a challenge term as part of the baby subspace assumption description, we have successfully simulated $\mathsf{Expt}^{\mathcal{MH}_j}_{R,m,L,N,T}$ (with the "corrupted" guards), and if we were given $\mathbf{C}_1$, we have successfully simulated $\mathsf{Expt}^{\mathcal{MH}_{j+1}}_{R,m,L,N,T}$ (again, with the "corrupted" guards), meaning our simulation is correct, which completes the proof.

8 Relaxations of Classic RKHwPRFs

8.1 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 8.1. (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 8.2. (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
$$F_{i,\ell}(k \otimes k', x) = \phi_{i,j,\ell}(F_{i,j}(k, x), F_{j+1,\ell}(k', x)).$$

8.1.1 Slotted RKHwPRF from SXDH

We show how to obtain a slotted RKHwPRF from any generic asymmetric multilinear map equipped with hardness of SXDH. 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
$$(h_1 = g^a_{\mathbf{v_1}}, h_2 = g^b_{\mathbf{v_2}}) \Rightarrow h_3 = g^{ab}_{\mathbf{v_1}+\mathbf{v_2}}$$
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, 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
$$H_{\mathbf{v}} := e(g^{\alpha_1}_{\mathbf{e}_1}, \ldots, g^{\alpha_N}_{\mathbf{e}_N}).$$

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.

8.1.2 Construction of iO 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 construction from classic RKHwPRFs from a functional point of view. We elaborate more on this below.

Multiplying in Order. Our iO construction only require elements from the output ring of a classic RKHwPRF (or more generally, an RHS) to be multiplied in a pre-determined order. 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.

Pre-Determined Multiplicative Depth. Our iO construction has a pre-fixed multiplicative depth. Concretely,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, 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 Strategy. Based on the aforementioned observations, our construction works from slotted RKHwPRFs in the same way as they work from classic RKHwPRFs. For example, intuitively, 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 argument for our iO construction from classic RKHwPRFs can be naturally adopted to work for the counterpart iO scheme 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 pseudorandomness) 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.

8.2 Slotted Partial RKHwPRFs

In this section, we define another flavor of slotted RKHwPRF called a slotted partial RKHwPRF.

Definition 8.3. (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 8.4. 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.

8.2.1 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 $e : \mathbb{G} \times \mathbb{G} \times \ldots \times \mathbb{G} \to \mathbb{G}_T$ 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 8.5. ($\mathcal{U}_{n,m}$-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 8.6. *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 8.7. *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 8.7 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. □

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 $i \leq j < \ell$, 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

$$\{F_{i,j} : (\mathbb{Z}_q^{m \times m} \times \mathcal{K}_i \times \mathcal{K}_{j+1}) \times \mathcal{X} \to \mathbb{G}_{i,j}\}_{i,j \in [N], i \leq j},$$

where for any $i, j \in [N]$ satisfying $i \leq j$, 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:

$$\begin{aligned} &F_{i,j}((\mathbf{K}_1, k_i, k_{j+1}), x) \cdot F_{i,j}((\mathbf{K}_2, k_i, k_{j+1}), x) \\ &\qquad = g_{i,j}^{H(k_i,x) \cdot \mathbf{K}_1 \cdot (H(k_{j+1},x))^{-1}} \cdot g_{i,j}^{H(k_i,x) \cdot \mathbf{K}_2 \cdot (H(k_{j+1},x))^{-1}} \\ &\qquad = g_{i,j}^{H(k_i,x) \cdot (\mathbf{K}_1 + \mathbf{K}_2) \cdot (H(k_{j+1},x))^{-1}} \\ &\qquad = F_{i,j}((\mathbf{K}_1 + \mathbf{K}_2, k_i, k_{j+1}), x). \end{aligned}$$

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:

$$\begin{aligned} &e(F_{i,j}((\mathbf{K}_1, k_i, k_{j+1}), x), F_{j+1,\ell}((\mathbf{K}_2, k_{j+1}, k_{\ell+1}), x)) \\ &\qquad = e\left(g_{i,j}^{H(k_i,x) \cdot \mathbf{K}_1 \cdot (H(k_{j+1},x))^{-1}} \quad , \quad g_{j+1,\ell}^{H(k_{j+1},x) \cdot \mathbf{K}_2 (H(k_{\ell+1},x))^{-1}}\right) \\ &\qquad = e(g_{i,j}, g_{j+1,\ell})^{H(k_i,x) \mathbf{K}_1 (H(k_{j+1},x))^{-1} \cdot H(k_{j+1},x) \cdot \mathbf{K}_2 \cdot (H(k_{\ell+1},x))^{-1}} \\ &\qquad = g_{i,\ell}^{H(k_i,x) \cdot \mathbf{K}_1 \cdot \mathbf{K}_2 \cdot (H(k_{\ell+1},x))^{-1}} \\ &\qquad = F_{i,\ell}((\mathbf{K}_1 \cdot \mathbf{K}_2, k_i, k_{\ell+1}), x). \end{aligned}$$

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 $n, m = \mathrm{poly}(\lambda)$, subject to the restriction that $n > m > N$. More formally, we state and prove the following lemma.

Lemma 8.8. *For any $i, j \in [N]$ such that $i \leq j$ and any arbitrarily large $L = \mathrm{poly}(\lambda)$, 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

$$\{x_\ell \leftarrow \mathcal{X}\}_{\ell \in [L]}, \quad (k_i, k_{j+1}) \leftarrow (\mathcal{K}_i \times \mathcal{K}_{j+1}), \quad \{\mathbf{K} \leftarrow \mathbb{Z}_q^{m \times m}\}_{\ell \in [L]}, \quad \{\mathbf{U}_{\ell,\ell'} \leftarrow \mathbb{Z}_q^{m \times m}\}_{\ell,\ell' \in [L]},$$

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 8.6 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

$$\begin{bmatrix} g_{i,j}^{\mathbf{U}_{1,0}^{(1)}\mathbf{A}_{1,1}^{-1}} & g_{i,j}^{\mathbf{A}_{1,0}\mathbf{K}_2\mathbf{A}_{1,1}^{-1}} & \cdots & g_{i,j}^{\mathbf{A}_{1,0}\mathbf{K}_L\mathbf{A}_{1,1}^{-1}} \\ \vdots & \vdots & \ddots & \vdots \\ g_{i,j}^{\mathbf{U}_{1,0}^{(L)}\mathbf{A}_{L,1}^{-1}} & g_{i,j}^{\mathbf{A}_{L,0}\mathbf{K}_2\mathbf{A}_{L,1}^{-1}} & \cdots & g_{i,j}^{\mathbf{A}_{L,0}\mathbf{K}_L\mathbf{A}_{L,1}^{-1}} \end{bmatrix}$$

$$\approx_s \begin{bmatrix} g_{i,j}^{\mathbf{U}_{1,1}} & g_{i,j}^{\mathbf{A}_{1,0}\mathbf{K}_2\mathbf{A}_{1,1}^{-1}} & \cdots & g_{i,j}^{\mathbf{A}_{1,0}\mathbf{K}_L\mathbf{A}_{1,1}^{-1}} \\ \vdots & \vdots & \ddots & \vdots \\ g_{i,j}^{\mathbf{U}_{L,1}} & g_{i,j}^{\mathbf{A}_{L,0}\mathbf{K}_2\mathbf{A}_{L,1}^{-1}} & \cdots & g_{i,j}^{\mathbf{A}_{L,0}\mathbf{K}_L\mathbf{A}_{L,1}^{-1}} \end{bmatrix},$$

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 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 8.9. *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

$$\begin{bmatrix} F_{1,1}((\mathbf{K}_{1,1}, (k_1, k_2)), x) & \cdots & F_{N,N}((\mathbf{K}_{1,N}, (k_N, k_{N+1})), x) \\ \vdots & \ddots & \vdots \\ F_{1,1}((\mathbf{K}_{L,1}, (k_1, k_2)), x) & \cdots & F_{N,N}((\mathbf{K}_{L,N}, (k_N, k_{N+1})), x) \end{bmatrix}$$

$$\approx_c \begin{bmatrix} g_{1,1}^{\mathbf{U}_{1,1}} & \cdots & g_{N,N}^{\mathbf{U}_{1,N}} \\ \vdots & \ddots & \vdots \\ g_{1,1}^{\mathbf{U}_{L,1}} & \cdots & g_{N,N}^{\mathbf{U}_{L,N}} \end{bmatrix}$$

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:

$$\left\{H_i(k_i, x)\right\}_{i \in [N+1]} \approx_c \left\{\mathbf{A}_i\right\}_{i \in [N+1]},$$

where for each $i \in [N + 1]$, $\mathbf{A}_i \leftarrow \mathbb{Z}_q^{m \times m}$. This in turn gives us the following:

$$\begin{bmatrix} F_{1,1}((\mathbf{K}_{1,1}, (k_1, k_2)), x) & \cdots & F_{N,N}((\mathbf{K}_{1,N}, (k_N, k_{N+1})), x) \\ \vdots & \ddots & \vdots \\ F_{1,1}((\mathbf{K}_{L,1}, (k_1, k_2)), x) & \cdots & F_{N,N}((\mathbf{K}_{L,N}, (k_N, k_{N+1})), x) \end{bmatrix}$$

$$\approx_c \begin{bmatrix} g_{1,1}^{\mathbf{A}_1\mathbf{K}_{1,1}\mathbf{A}_2^{-1}} & \cdots & g_{N,N}^{\mathbf{A}_N\mathbf{K}_{1,N}\mathbf{A}_{N+1}^{-1}} \\ \vdots & \ddots & \vdots \\ g_{1,1}^{\mathbf{A}_1\mathbf{K}_{L,1}\mathbf{A}_2^{-1}} & \cdots & g_{N,N}^{\mathbf{A}_N\mathbf{K}_{L,N}\mathbf{A}_{N+1}^{-1}} \end{bmatrix},$$

where for each $i \in [N+1]$, $\mathbf{A}_i \leftarrow \mathbb{Z}_q^{m\times m}$. Next, we invoke Lemma 8.7 to argue

$$\begin{bmatrix} g^{\mathbf{A}_1\mathbf{K}_{1,1}\mathbf{A}_2^{-1}} & \cdots & g^{\mathbf{A}_N\mathbf{K}_{1,N}\mathbf{A}_{N+1}^{-1}} \\ \vdots & \ddots & \vdots \\ g^{\mathbf{A}_1\mathbf{K}_{L,1}\mathbf{A}_2^{-1}} & \cdots & g^{\mathbf{A}_N\mathbf{K}_{L,N}\mathbf{A}_{N+1}^{-1}} \end{bmatrix}$$

$$\approx_s \begin{bmatrix} g_{1,1}^{\mathbf{U}_{1,1}} & \cdots & g_{N,N}^{\mathbf{U}_{1,N}} \\ \vdots & \ddots & \vdots \\ g_{1,1}^{\mathbf{U}_{L,1}} & \cdots & g_{N,N}^{\mathbf{U}_{L,N}} \end{bmatrix}.$$

This completes the proof of parallel input security. □

8.2.2 Construction of iO from Slotted Partial RKHwPRFs

Note that our construction of iO 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 8.1.2 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 differences between slotted RKHwPRFs and slotted partial 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 8.3). 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 Homomorphic 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 our iO construction also has a pre-fixed additive depth. Note that 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. 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 construction, the correctness and security guarantees translate naturally.

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.

8.3 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. To our knowledge, it has not been broken or attacked in any significant way. However, before this work (again, to our knowledge) it was not known to imply iO.

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 in Figure 18. We avoid formal descriptions of these functions, which essentially follow from the informal description presented in Section 2.7. The readers may refer [MZ18] for the formal descriptions. We subsequently show an explicit construction of slotted partial RKHwPRF from the MZ18 MMap.

Choosing the Computational Assumption. We now focus on choosing the appropriate computational assumption over the MZ18 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 $T = \mathrm{poly}(\lambda)$:

$$\begin{aligned}(\mathsf{pp}, \{\mathsf{Enc\text{-}MZ}(\mathsf{pp}, a_t, (i,j); r_t), \mathsf{Enc\text{-}MZ}(\mathsf{pp}, k \cdot a_t, (i,j); r'_t)\}_{t\in[T]}) \\ \approx_c (\mathsf{pp}, \{\mathsf{Enc\text{-}MZ}(\mathsf{pp}, a_t, (i,j); r_t), \mathsf{Enc\text{-}MZ}(\mathsf{pp}, b_t, (i,j); r''_t)\}_{t\in[T]}),\end{aligned}$$

where $k, a_1, \ldots, a_T, b_1, \ldots, b_T$ are uniformly sampled plaintext elements in $\mathbb{Z}_M$, and r_t, r'_t, r''_t for $t \in [T]$ are appropriately distributed random coins for the MZ18 encoding algorithm. Note the "standard" presentation of the SXDH assumption is the above when $T = 2$.

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

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.

Figure 18: The MZ18 MMap from [MZ18]

MZ18 encoding of a at interval-level $[i,j]$. As a result, this assumption is not immediately useful to construct weak PRF families. Note that this is also why we parameterize the assumption with T rather than set $T = 2$: we might need more terms and we cannot do traditional "DDH-style" rerandomization.

We overcome this difficulty by using the following alternative computational assumption called "shifted"-SXDH: letting $\mathsf{pp} = \mathsf{Setup\text{-}MZ}(1^\lambda, N)$, we assume that the following ensembles are computationally indistinguishable for any index $i \in [N]$ and any $T = \mathrm{poly}(\lambda)$:

$$\begin{aligned}(\mathsf{pp}, \{\mathsf{Enc\text{-}MZ}(\mathsf{pp}, a_t, (i,j); r_t), \mathsf{Enc\text{-}MZ}(\mathsf{pp}, k \cdot a_t, (i, j+1); r'_t)\}_{t \in [T]}) \\ \approx_c (\mathsf{pp}, \{\mathsf{Enc\text{-}MZ}(\mathsf{pp}, a_t, (i,j); r_t), \mathsf{Enc\text{-}MZ}(\mathsf{pp}, b_t, (i, j+1); r''_t)\}_{t \in [T]}),\end{aligned}$$

where $k, a_1, \ldots, a_T, b_1, \ldots, b_T$ are uniformly sampled plaintext elements in $\mathbb{Z}_M$, and r_t, r'_t, r''_t for $t \in [T]$ 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. We state this in a very simple lemma.

Lemma 8.10. *Any adversary that can break the "shifted"–SXDH assumption over the MZ18 multilinear map can be used to break the regular SXDH assumption over the same map.*

Proof. 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. More precisely, the PPT algorithm challenger just takes the SXDH instance and multiplies the encodings containing the scalar k by encodings of 1 at interval $[j+1, j+1]$. □

In other words, we can still base the security of our RKHwPRF family on the SXDH assumption over the MZ18 multilinear map.

Constructing Partial Slotted Weak PRFs. Equipped with the aforementioned computational assumption, we now show how to concretely construct a partial slotted weak PRF family from the MZ18 MMap. Before we get started, we will outline our approach. We will use a series of hash functions $H_{i,j}$ (that we will formally define shortly) that map arbitrary input to elements encoded in the interval $[i, j]$. These functions are defined in a somewhat contrived manner because we need the key homomorphism (i.e., the homomorphism with respect to the addition and multiplication operations between the key-space and output-space for the same input) to be consistent across different levels.

To compute the (weak) PRF, we "pair" these encoded elements corresponding to the inputs with an element k (representing the PRF key) encoded at an adjacent level. Note that due to certain structural restrictions inherent to the MZ18 MMap construction, this requires us to use two levels of the multilinear map in order to encode a single element. We elaborate on this in more detail later in this section.

We now describe the weak PRF construction formally. Let pp be public parameters generated as $\mathsf{pp} = \mathsf{Setup\text{-}MZ}(1^\lambda, 2N+1)$. Let $\mathcal{X}$ be a set of size M and assume that there exists a family of (weakly pseudorandom) 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 $r_{i,j'}$ such that the following relation holds:

$$H_{i,j'}(x) = \mathsf{Mult\text{-}MZ}(\mathsf{pp}, H_{i,j}(x), H_{j+1,j'}(x), r_{i,j'}).$$

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:

  $$H_{i,j'}(x) = \mathsf{Mult\text{-}MZ}(H_{i,j}(x), H_{j+1,j'}(x), r_{i,j'}),$$

  where $r_{i,j'}$ 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.

[1] We implicitly assume that the MZ18 public parameter pp is part of the PRF family description.

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.

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-embedded 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.

9 Conclusion and Future Work

In this paper, we showed how to build iO 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.

Acknowledgement

We thank Navid Alamati for helpful discussions on an earlier version of this paper. We also thank the anonymous reviewers of IACR CIC 2025 for their patient and careful reading, and for their constructive comments and feedback.

References

[AB15] Benny Applebaum and Zvika Brakerski. Obfuscating circuits via composite-order graded encoding. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015: 12th Theory of Cryptography Conference, Part II*, volume 9015 of *Lecture Notes in Computer Science*, pages 528–556, Warsaw, Poland, March 23–25, 2015. Springer Berlin Heidelberg, Germany. doi:10.1007/978-3-662-46497-7_21.

[AFH+16] Martin R. Albrecht, Pooya Farshim, Dennis Hofheinz, Enrique Larraia, and Kenneth G. Paterson. Multilinear maps from obfuscation. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A: 13th Theory of Cryptography Conference, Part I*, volume 9562 of *Lecture Notes in Computer Science*, pages 446–473, Tel Aviv, Israel, January 10–13, 2016. Springer Berlin Heidelberg, Germany. doi:10.1007/978-3-662-49096-9_19.

[AFH+20] Martin R. Albrecht, Pooya Farshim, Shuai Han, Dennis Hofheinz, Enrique Larraia, and Kenneth G. Paterson. Multilinear maps from obfuscation. *Journal of Cryptology*, 33(3):1080–1113, July 2020. doi:10.1007/s00145-019-09340-0.

[Agr19] Shweta Agrawal. Indistinguishability obfuscation without multilinear maps: New methods for bootstrapping and instantiation. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2019, Part I*, volume 11476 of *Lecture Notes in Computer Science*, pages 191–225, Darmstadt, Germany, May 19–23, 2019. Springer, Cham, Switzerland. doi:10.1007/978-3-030-17653-2_7.

[AJ15] Prabhanjan Ananth and Abhishek Jain. Indistinguishability obfuscation from compact functional encryption. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *Advances in Cryptology – CRYPTO 2015, Part I*, volume 9215 of *Lecture Notes in Computer Science*, pages 308–326, Santa Barbara, CA, USA, August 16–20, 2015. Springer Berlin Heidelberg, Germany. doi:10.1007/978-3-662-47989-6_15.

[AJL+19] Prabhanjan Ananth, Aayush Jain, Huijia Lin, Christian Matt, and Amit Sahai. Indistinguishability obfuscation without multilinear maps: New paradigms via low degree weak pseudorandomness and security amplification. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019, Part III*, volume 11694 of *Lecture Notes in Computer Science*, pages 284–332, Santa Barbara, CA, USA, August 18–22, 2019. Springer, Cham, Switzerland. doi:10.1007/978-3-030-26954-8_10.

[Alb19] Martin Albrecht. Are graded encoding schemes broken yet?, 2019. URL: https://malb.io/are-graded-encoding-schemes-broken-yet.html.

[Ale03] Michael Alekhnovich. More on average case vs approximation complexity. In *44th Annual Symposium on Foundations of Computer Science*, pages 298–307, Cambridge, MA, USA, October 11–14, 2003. IEEE Computer Society Press. doi:10.1109/SFCS.2003.1238204.

[AM18] Shweta Agrawal and Monosij Maitra. FE and iO for Turing machines from minimal assumptions. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018: 16th Theory of Cryptography Conference, Part II*, volume 11240 of *Lecture Notes in Computer Science*, pages 473–512, Panaji, India, November 11–14, 2018. Springer, Cham, Switzerland. doi:10.1007/978-3-030-03810-6_18.

[AMP19] Navid Alamati, Hart Montgomery, and Sikhar Patranabis. Symmetric primitives with structured secrets. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019, Part I*, volume 11692 of *Lecture Notes in Computer Science*, pages 650–679, Santa Barbara, CA, USA, August 18–22, 2019. Springer, Cham, Switzerland. doi:10.1007/978-3-030-26948-7_23.

[AMP23] Navid Alamati, Hart Montgomery, and Sikhar Patranabis. Multiparty noninteractive key exchange from ring key-homomorphic weak prfs. In *CT-RSA 2023*, volume 13871, pages 333–359, 2023. doi:10.1007/978-3-031-30872-7_13.

[AMPR19] Navid Alamati, Hart Montgomery, Sikhar Patranabis, and Arnab Roy. Minicrypt primitives with algebraic structure and applications. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2019, Part II*, volume 11477 of *Lecture Notes in Computer Science*, pages 55–82, Darmstadt, Germany, May 19–23, 2019. Springer, Cham, Switzerland. doi:10.1007/978-3-030-17656-3_3.

[AS17] Prabhanjan Ananth and Amit Sahai. Projective arithmetic functional encryption and indistinguishability obfuscation from degree-5 multilinear maps. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology – EUROCRYPT 2017, Part I*, volume 10210 of *Lecture Notes in Computer Science*, pages 152–181, Paris, France, April 30 – May 4, 2017. Springer, Cham, Switzerland. doi:10.1007/978-3-319-56620-7_6.

[Bar86] David A. Mix Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1. In *18th Annual ACM Symposium on Theory of Computing*, pages 1–5, Berkeley, CA, USA, May 28–30, 1986. ACM Press. doi:10.1145/12130.12131.

[Bar17] Boaz Barak. The complexity of public-key cryptography. In Yehuda Lindell, editor, *Tutorials on the Foundations of Cryptography*, pages 45–77. Springer International Publishing, 2017. doi:10.1007/978-3-319-57048-8_2.

[BBKK18] Boaz Barak, Zvika Brakerski, Ilan Komargodski, and Pravesh K. Kothari. Limits on low-degree pseudorandom generators (or: Sum-of-squares meets program obfuscation). In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018, Part II*, volume 10821 of *Lecture Notes in Computer Science*, pages 649–679, Tel Aviv, Israel, April 29 – May 3, 2018. Springer, Cham, Switzerland. doi:10.1007/978-3-319-78375-8_21.

[BDGM20a] Zvika Brakerski, Nico Döttling, Sanjam Garg, and Giulio Malavolta. Candidate iO from homomorphic encryption schemes. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology – EUROCRYPT 2020, Part I*, volume 12105 of *Lecture Notes in Computer Science*, pages 79–109, Zagreb, Croatia, May 10–14, 2020. Springer, Cham, Switzerland. doi:10.1007/978-3-030-45721-1_4.

[BDGM20b] Zvika Brakerski, Nico Döttling, Sanjam Garg, and Giulio Malavolta. Factoring and pairings are not necessary for io: Circular-secure lwe suffices. Cryptology ePrint Archive, Report 2020/1024, 2020. https://eprint.iacr.org/2020/1024.

[BF01] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229, Santa Barbara, CA, USA, August 19–23, 2001. Springer Berlin Heidelberg, Germany. doi:10.1007/3-540-44647-8_13.

[BGG+18] Dan Boneh, Rosario Gennaro, Steven Goldfeder, Aayush Jain, Sam Kim, Peter M. R. Rasmussen, and Amit Sahai. Threshold cryptosystems from threshold fully homomorphic encryption. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018, Part I*, volume 10991 of *Lecture Notes in Computer Science*, pages 565–596, Santa Barbara, CA, USA, August 19–23, 2018. Springer, Cham, Switzerland. doi:10.1007/978-3-319-96884-1_19.

[BGI+01] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In Joe Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 1–18, Santa Barbara, CA, USA, August 19–23, 2001. Springer Berlin Heidelberg, Germany. doi:10.1007/3-540-44647-8_1.

[BGK+14] Boaz Barak, Sanjam Garg, Yael Tauman Kalai, Omer Paneth, and Amit Sahai. Protecting obfuscation against algebraic attacks. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology – EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 221–238, Copenhagen, Denmark, May 11–15, 2014. Springer Berlin Heidelberg, Germany. doi:10.1007/978-3-642-55220-5_13.

[BGMZ18] James Bartusek, Jiaxin Guan, Fermi Ma, and Mark Zhandry. Return of GGH15: Provable security against zeroizing attacks. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018: 16th Theory of Cryptography Conference, Part II*, volume 11240 of *Lecture Notes in Computer Science*, pages 544–574, Panaji, India, November 11–14, 2018. Springer, Cham, Switzerland. doi:10.1007/978-3-030-03810-6_20.

[BKM17] Dan Boneh, Sam Kim, and Hart William Montgomery. Private puncturable PRFs from standard lattice assumptions. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology – EUROCRYPT 2017, Part I*, volume 10210 of *Lecture Notes in Computer Science*, pages 415–445, Paris, France, April 30 – May 4, 2017. Springer, Cham, Switzerland. doi:10.1007/978-3-319-56620-7_15.

[BKM+19] Allison Bishop, Lucas Kowalczyk, Tal Malkin, Valerio Pastro, Mariana Raykova, and Kevin Shi. In pursuit of clarity in obfuscation. Cryptology ePrint Archive, Report 2019/463, 2019. https://eprint.iacr.org/2019/463.

[BLMR13] Dan Boneh, Kevin Lewi, Hart William Montgomery, and Ananth Raghunathan. Key homomorphic PRFs and their applications. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 410–428, Santa Barbara, CA, USA, August 18–22, 2013. Springer Berlin Heidelberg, Germany. doi:10.1007/978-3-642-40041-4_23.

[BLS01] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. In Colin Boyd, editor, *Advances in Cryptology – ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 514–532, Gold Coast, Australia, December 9–13, 2001. Springer Berlin Heidelberg, Germany. doi:10.1007/3-540-45682-1_30.

[BMSZ15] Saikrishna Badrinarayanan, Eric Miles, Amit Sahai, and Mark Zhandry. Post-zeroizing obfuscation: The case of evasive circuits. Cryptology ePrint Archive, Report 2015/167, 2015. URL: https://eprint.iacr.org/2015/167.

[BP15] Nir Bitansky and Omer Paneth. ZAPs and non-interactive witness indistinguishability from indistinguishability obfuscation. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015: 12th Theory of Cryptography Conference, Part II*, volume 9015 of *Lecture Notes in Computer Science*, pages 401–427, Warsaw, Poland, March 23–25, 2015. Springer Berlin Heidelberg, Germany. doi:10.1007/978-3-662-46497-7_16.

[BR14] Zvika Brakerski and Guy N. Rothblum. Virtual black-box obfuscation for all circuits via generic graded encoding. In Yehuda Lindell, editor, *TCC 2014: 11th Theory of Cryptography Conference*, volume 8349 of *Lecture Notes in Computer Science*, pages 1–25, San Diego, CA, USA, February 24–26, 2014. Springer Berlin Heidelberg, Germany. doi:10.1007/978-3-642-54242-8_1.

[BS89] Ravi B Boppana and Michael Sipser. *The complexity of finite functions*, volume 99. Laboratory for Computer Science, Massachusetts Institute of Technology, 1989. doi:10.1016/B978-0-444-88071-0.50019-9.

[BSW11] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In Yuval Ishai, editor, *TCC 2011: 8th Theory of Cryptography Conference*, volume 6597 of *Lecture Notes in Computer Science*, pages 253–273, Providence, RI, USA, March 28–30, 2011. Springer Berlin Heidelberg, Germany. doi:10.1007/978-3-642-19571-6_16.

[BV15] Nir Bitansky and Vinod Vaikuntanathan. Indistinguishability obfuscation from functional encryption. In Venkatesan Guruswami, editor, *56th Annual Symposium on Foundations of Computer Science*, pages 171–190, Berkeley, CA, USA, October 17–20, 2015. IEEE Computer Society Press. doi:10.1109/FOCS.2015.20.

[BZ14] Dan Boneh and Mark Zhandry. Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014, Part I*, volume 8616 of *Lecture Notes in Computer Science*, pages 480–499, Santa Barbara, CA, USA, August 17–21, 2014. Springer Berlin Heidelberg, Germany. doi:10.1007/978-3-662-44371-2_27.

[CC17] Ran Canetti and Yilei Chen. Constraint-hiding constrained PRFs for NC^1 from LWE. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology – EUROCRYPT 2017, Part I*, volume 10210 of *Lecture Notes in Computer Science*, pages 446–476, Paris, France, April 30 – May 4, 2017. Springer, Cham, Switzerland. doi:10.1007/978-3-319-56620-7_16.

[CCH+19] Jung Hee Cheon, Wonhee Cho, Minki Hhan, Jiseung Kim, and Changmin Lee. Statistical zeroizing attack: Cryptanalysis of candidates of BP obfuscation over GGH15 multilinear map. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019, Part III*, volume 11694 of *Lecture Notes in Computer Science*, pages 253–283, Santa Barbara, CA, USA, August 18–22, 2019. Springer, Cham, Switzerland. doi:10.1007/978-3-030-26954-8_9.

[CGH17] Yilei Chen, Craig Gentry, and Shai Halevi. Cryptanalyses of candidate branching program obfuscators. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology – EUROCRYPT 2017, Part III*, volume 10212 of *Lecture Notes in Computer Science*, pages 278–307, Paris, France, April 30 – May 4, 2017. Springer, Cham, Switzerland. doi:10.1007/978-3-319-56617-7_10.

[CHL+15] Jung Hee Cheon, Kyoohyung Han, Changmin Lee, Hansol Ryu, and Damien Stehlé. Cryptanalysis of the multilinear map over the integers. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015, Part I*, volume 9056 of *Lecture Notes in Computer Science*, pages 3–12, Sofia, Bulgaria, April 26–30, 2015. Springer Berlin Heidelberg, Germany. doi:10.1007/978-3-662-46800-5_1.

[CLLT16] Jean-Sébastien Coron, Moon Sung Lee, Tancrède Lepoint, and Mehdi Tibouchi. Cryptanalysis of GGH15 multilinear maps. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016, Part II*, volume 9815 of *Lecture Notes in Computer Science*, pages 607–628, Santa Barbara, CA, USA, August 14–18, 2016. Springer Berlin Heidelberg, Germany. doi:10.1007/978-3-662-53008-5_21.

[CLLT17] Jean-Sébastien Coron, Moon Sung Lee, Tancrède Lepoint, and Mehdi Tibouchi. Zeroizing attacks on indistinguishability obfuscation over CLT13. In Serge Fehr, editor, *PKC 2017: 20th International Conference on Theory and Practice of Public Key Cryptography, Part I*, volume 10174 of *Lecture Notes in Computer Science*, pages 41–58, Amsterdam, The Netherlands, March 28–31, 2017. Springer Berlin Heidelberg, Germany. doi:10.1007/978-3-662-54365-8_3.

[CLT13] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. Practical multilinear maps over the integers. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 476–493, Santa Barbara, CA, USA, August 18–22, 2013. Springer Berlin Heidelberg, Germany. doi:10.1007/978-3-642-40041-4_26.

[CLT15] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. New multilinear maps over the integers. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *Advances in Cryptology – CRYPTO 2015, Part I*, volume 9215 of *Lecture Notes in Computer Science*, pages 267–286, Santa Barbara, CA, USA, August 16–20, 2015. Springer Berlin Heidelberg, Germany. doi:10.1007/978-3-662-47989-6_13.

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Trans. Inf. Theory*, 22(6):644–654, 1976. doi:10.1109/TIT.1976.1055638.

[DJ01] Ivan Damgård and Mats Jurik. A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In Kwangjo Kim, editor, *PKC 2001: 4th International Workshop on Theory and Practice in Public Key Cryptography*, volume 1992 of *Lecture Notes in Computer Science*, pages 119–136, Cheju Island, South Korea, February 13–15, 2001. Springer Berlin Heidelberg, Germany. doi:10.1007/3-540-44586-2_9.

[DQV+21] Lalita Devadas, Willy Quach, Vinod Vaikuntanathan, Hoeteck Wee, and Daniel Wichs. Succinct LWE sampling, random polynomials, and obfuscation. In Kobbi Nissim and Brent Waters, editors, *TCC 2021: 19th Theory of Cryptography Conference, Part II*, volume 13043 of *Lecture Notes in Computer Science*, pages 256–287, Raleigh, NC, USA, November 8–11, 2021. Springer, Cham, Switzerland. doi:10.1007/978-3-030-90453-1_9.

[EHK+13] Alex Escala, Gottfried Herold, Eike Kiltz, Carla Ràfols, and Jorge Villar. An algebraic framework for Diffie-Hellman assumptions. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part II*, volume 8043 of *Lecture Notes in Computer Science*, pages 129–147, Santa Barbara, CA, USA, August 18–22, 2013. Springer Berlin Heidelberg, Germany. doi:10.1007/978-3-642-40084-1_8.

[FHHL18] Pooya Farshim, Julia Hesse, Dennis Hofheinz, and Enrique Larraia. Graded encoding schemes from obfuscation. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018: 21st International Conference on Theory and Practice of Public Key Cryptography, Part II*, volume 10770 of *Lecture Notes in Computer Science*, pages 371–400, Rio de Janeiro, Brazil, March 25–29, 2018. Springer, Cham, Switzerland. doi:10.1007/978-3-319-76581-5_13.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *41st Annual ACM Symposium on Theory of Computing*, pages 169–178, Bethesda, MD, USA, May 31 – June 2, 2009. ACM Press. doi:10.1145/1536414.1536440.

[GGH13a] Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology – EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 1–17, Athens, Greece, May 26–30, 2013. Springer Berlin Heidelberg, Germany. doi:10.1007/978-3-642-38348-9_1.

[GGH+13b] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th Annual Symposium on Foundations of Computer Science*, pages 40–49, Berkeley, CA, USA, October 26–29, 2013. IEEE Computer Society Press. doi:10.1109/FOCS.2013.13.

[GGH15] Craig Gentry, Sergey Gorbunov, and Shai Halevi. Graph-induced multilinear maps from lattices. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015: 12th Theory of Cryptography Conference, Part II*, volume 9015 of *Lecture Notes in Computer Science*, pages 498–527, Warsaw, Poland, March 23–25, 2015. Springer Berlin Heidelberg, Germany. doi:10.1007/978-3-662-46497-7_20.

[GGHR14] Sanjam Garg, Craig Gentry, Shai Halevi, and Mariana Raykova. Two-round secure MPC from indistinguishability obfuscation. In Yehuda Lindell, editor, *TCC 2014: 11th Theory of Cryptography Conference*, volume 8349 of *Lecture Notes in Computer Science*, pages 74–94, San Diego, CA, USA, February 24–26, 2014. Springer Berlin Heidelberg, Germany. doi:10.1007/978-3-642-54242-8_4.

[GJLS21] Romain Gay, Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from simple-to-state hard problems: New assumptions, new techniques, and simplification. In Anne Canteaut and François-Xavier Standaert, editors, *Advances in Cryptology – EUROCRYPT 2021, Part III*, volume 12698 of *Lecture Notes in Computer Science*, pages 97–126, Zagreb, Croatia, October 17–21, 2021. Springer, Cham, Switzerland. doi:10.1007/978-3-030-77883-5_4.

[GLSW15] Craig Gentry, Allison Bishop Lewko, Amit Sahai, and Brent Waters. Indistinguishability obfuscation from the multilinear subgroup elimination assumption. In Venkatesan Guruswami, editor, *56th Annual Symposium on Foundations of Computer Science*, pages 151–170, Berkeley, CA, USA, October 17–20, 2015. IEEE Computer Society Press. doi:10.1109/FOCS.2015.19.

[GMM+16] Sanjam Garg, Eric Miles, Pratyay Mukherjee, Amit Sahai, Akshayaram Srinivasan, and Mark Zhandry. Secure obfuscation in a weak multilinear map model. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B: 14th Theory of Cryptography Conference, Part II*, volume 9986 of *Lecture Notes in Computer Science*, pages 241–268, Beijing, China, October 31 – November 3, 2016. Springer Berlin Heidelberg, Germany. doi:10.1007/978-3-662-53644-5_10.

[GMM17] Sanjam Garg, Mohammad Mahmoody, and Ameer Mohammed. Lower bounds on obfuscation from all-or-nothing encryption primitives. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part I*, volume 10401 of *Lecture Notes in Computer Science*, pages 661–695, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Cham, Switzerland. doi:10.1007/978-3-319-63688-7_22.

[GP21] Romain Gay and Rafael Pass. Indistinguishability obfuscation from circular security. In *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 736–749. ACM, 2021. doi:10.1145/3406325.3451070.

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 75–92, Santa Barbara, CA, USA, August 18–22, 2013. Springer Berlin Heidelberg, Germany. doi:10.1007/978-3-642-40041-4_5.

[HB15] Máté Horváth and Levente Buttyán. The birth of cryptographic obfuscation – a survey. Cryptology ePrint Archive, Paper 2015/412, 2015. URL: https://eprint.iacr.org/2015/412.

[HJ16] Yupu Hu and Huiwen Jia. Cryptanalysis of GGH map. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology – EUROCRYPT 2016, Part I*, volume 9665 of *Lecture Notes in Computer Science*, pages 537–565, Vienna, Austria, May 8–12, 2016. Springer Berlin Heidelberg, Germany. doi:10.1007/978-3-662-49890-3_21.

[HJL21] Samuel B. Hopkins, Aayush Jain, and Huijia Lin. Counterexamples to new circular security assumptions underlying iO. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology – CRYPTO 2021, Part II*, volume 12826 of *Lecture Notes in Computer Science*, pages 673–700, Virtual Event, August 16–20, 2021. Springer, Cham, Switzerland. doi:10.1007/978-3-030-84245-1_23.

[HSW14] Susan Hohenberger, Amit Sahai, and Brent Waters. Replacing a random oracle: Full domain hash from indistinguishability obfuscation. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology – EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 201–220, Copenhagen, Denmark, May 11–15, 2014. Springer Berlin Heidelberg, Germany. doi:10.1007/978-3-642-55220-5_12.

[IZ89] Russell Impagliazzo and David Zuckerman. How to recycle random bits. In *30th Annual Symposium on Foundations of Computer Science*, pages 248–253, Research Triangle Park, NC, USA, October 30 – November 1, 1989. IEEE Computer Society Press. doi:10.1109/SFCS.1989.63486.

[JLLS23] Aayush Jain, Huijia Lin, Paul Lou, and Amit Sahai. Polynomial-time cryptanalysis of the subspace flooding assumption for post-quantum $i\mathcal{O}$. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology – EUROCRYPT 2023, Part I*, volume 14004 of *Lecture Notes in Computer Science*, pages 205–235, Lyon, France, April 23–27, 2023. Springer, Cham, Switzerland. doi:10.1007/978-3-031-30545-0_8.

[JLMS19] Aayush Jain, Huijia Lin, Christian Matt, and Amit Sahai. How to leverage hardness of constant-degree expanding polynomials over $\mathbb{R}$ to build $i\mathcal{O}$. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2019, Part I*, volume 11476 of *Lecture Notes in Computer Science*, pages 251–281, Darmstadt, Germany, May 19–23, 2019. Springer, Cham, Switzerland. doi:10.1007/978-3-030-17653-2_9.

[JLS21] Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from well-founded assumptions. In Samir Khuller and Virginia Vassilevska Williams, editors, *53rd Annual ACM Symposium on Theory of Computing*, pages 60–73, Virtual Event, Italy, June 21–25, 2021. ACM Press. doi:10.1145/3406325.3451093.

[JLS22] Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from LPN over $\mathbb{F}_p$, DLIN, and PRGs in NC^0. In Orr Dunkelman and Stefan Dziembowski, editors, *Advances in Cryptology – EUROCRYPT 2022, Part I*, volume 13275 of *Lecture Notes in Computer Science*, pages 670–699, Trondheim, Norway, May 30 – June 3, 2022. Springer, Cham, Switzerland. doi:10.1007/978-3-031-06944-4_23.

[Jou04] Antoine Joux. A one round protocol for tripartite diffie-hellman. *J. Cryptol.*, 17(4):263–276, 2004. URL: https://doi.org/10.1007/s00145-004-0312-y, doi:10.1007/S00145-004-0312-Y.

[Kil88] Joe Kilian. Founding cryptography on oblivious transfer. In *20th Annual ACM Symposium on Theory of Computing*, pages 20–31, Chicago, IL, USA, May 2–4, 1988. ACM Press. doi:10.1145/62212.62215.

[Lin16] Huijia Lin. Indistinguishability obfuscation from constant-degree graded encoding schemes. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology – EUROCRYPT 2016, Part I*, volume 9665 of *Lecture Notes in Computer Science*, pages 28–57, Vienna, Austria, May 8–12, 2016. Springer Berlin Heidelberg, Germany. doi:10.1007/978-3-662-49890-3_2.

[Lin17] Huijia Lin. Indistinguishability obfuscation from SXDH on 5-linear maps and locality-5 PRGs. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part I*, volume 10401 of *Lecture Notes in Computer Science*, pages 599–629, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Cham, Switzerland. doi:10.1007/978-3-319-63688-7_20.

[LT17] Huijia Lin and Stefano Tessaro. Indistinguishability obfuscation from trilinear maps and block-wise local PRGs. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part I*, volume 10401 of *Lecture Notes in Computer Science*, pages 630–660, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Cham, Switzerland. doi:10.1007/978-3-319-63688-7_21.

[LV16] Huijia Lin and Vinod Vaikuntanathan. Indistinguishability obfuscation from DDH-like assumptions on constant-degree graded encodings. In Irit Dinur, editor, *57th Annual Symposium on Foundations of Computer Science*, pages 11–20, New Brunswick, NJ, USA, October 9–11, 2016. IEEE Computer Society Press. doi:10.1109/FOCS.2016.11.

[LV17] Alex Lombardi and Vinod Vaikuntanathan. Limits on the locality of pseudorandom generators and applications to indistinguishability obfuscation. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017: 15th Theory of Cryptography Conference, Part I*, volume 10677 of *Lecture Notes in Computer Science*, pages 119–137, Baltimore, MD, USA, November 12–15, 2017. Springer, Cham, Switzerland. doi:10.1007/978-3-319-70500-2_5.

[Mic02] Daniele Micciancio. Generalized compact knapsacks, cyclic lattices, and efficient one-way functions from worst-case complexity assumptions. In *43rd Annual Symposium on Foundations of Computer Science*, pages 356–365, Vancouver, BC, Canada, November 16–19, 2002. IEEE Computer Society Press. doi:10.1109/SFCS.2002.1181960.

[MSZ16] Eric Miles, Amit Sahai, and Mark Zhandry. Annihilation attacks for multilinear maps: Cryptanalysis of indistinguishability obfuscation over GGH13. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016, Part II*, volume 9815 of *Lecture Notes in Computer Science*, pages 629–658, Santa Barbara, CA, USA, August 14–18, 2016. Springer Berlin Heidelberg, Germany. doi:10.1007/978-3-662-53008-5_22.

[MZ18] Fermi Ma and Mark Zhandry. The MMap strikes back: Obfuscation and new multilinear maps immune to CLT13 zeroizing attacks. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018: 16th Theory of Cryptography Conference, Part II*, volume 11240 of *Lecture Notes in Computer Science*, pages 513–543, Panaji, India, November 11–14, 2018. Springer, Cham, Switzerland. doi:10.1007/978-3-030-03810-6_19.

[NPR99] Moni Naor, Benny Pinkas, and Omer Reingold. Distributed pseudo-random functions and KDCs. In Jacques Stern, editor, *Advances in Cryptology – EUROCRYPT'99*, volume 1592 of *Lecture Notes in Computer Science*, pages 327–346, Prague, Czech Republic, May 2–6, 1999. Springer Berlin Heidelberg, Germany. doi:10.1007/3-540-48910-X_23.

[NR95] Moni Naor and Omer Reingold. Synthesizers and their application to the parallel construction of pseudo-random functions. In *36th Annual Symposium on Foundations of Computer Science*, pages 170–181, Milwaukee, Wisconsin, October 23–25, 1995. IEEE Computer Society Press. doi:10.1109/SFCS.1995.492474.

[Rao20] Anup Rao. Lecture 2: Barrington's theorem, 2020. URL: https://homes.cs.washington.edu/~anuprao/pubs/CSE531Sp2020/lecture2.pdf.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th Annual ACM Symposium on Theory of Computing*, pages 84–93, Baltimore, MA, USA, May 22–24, 2005. ACM Press. doi:10.1145/1060590.1060603.

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, 1978. doi:10.1145/359340.359342.

[RVV24] Seyoon Ragavan, Neekon Vafa, and Vinod Vaikuntanathan. Indistinguishability obfuscation from bilinear maps and LPN variants. In Elette Boyle and Mohammad Mahmoody, editors, *TCC 2024: 22nd Theory of Cryptography Conference, Part IV*, volume 15367 of *Lecture Notes in Computer Science*, pages 3–36, Milan, Italy, December 2–6, 2024. Springer, Cham, Switzerland. doi:10.1007/978-3-031-78023-3_1.

[Sha79] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979. doi:10.1145/359168.359176.

[SW14] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In David B. Shmoys, editor, *46th Annual ACM Symposium on Theory of Computing*, pages 475–484, New York, NY, USA, May 31 – June 3, 2014. ACM Press. doi:10.1145/2591796.2591825.

[WW21] Hoeteck Wee and Daniel Wichs. Candidate obfuscation via oblivious LWE sampling. In Anne Canteaut and François-Xavier Standaert, editors, *Advances in Cryptology - EUROCRYPT 2021 - 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Proceedings, Part III*, volume 12698 of *Lecture Notes in Computer Science*, pages 127–156. Springer, 2021. doi:10.1007/978-3-030-77883-5_5.

[YYHK20] Takashi Yamakawa, Shota Yamada, Goichiro Hanaoka, and Noboru Kunihiro. Generic hardness of inversion on ring and its relation to self-bilinear map. *Theor. Comput. Sci.*, 820:60–84, 2020. doi:10.1016/J.TCS.2020.03.009.

[Zim15] Joe Zimmerman. How to obfuscate programs directly. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015, Part II*, volume 9057 of *Lecture Notes in Computer Science*, pages 439–467, Sofia, Bulgaria, April 26–30, 2015. Springer Berlin Heidelberg, Germany. doi:10.1007/978-3-662-46803-6_15.

A Some Basic Lemmata

In this section, we prove a number of useful facts about any ring R such that for any $m, Q = \mathrm{poly}(\lambda)$ where m is any arbitrary positive integer satisfying $m > 3\lambda$ and Q is any arbitrary positive integer, we have

$$(\mathbf{A}, \mathbf{AK}) \overset{c}{\approx} (\mathbf{A}, \mathbf{U}), \quad (\mathbf{B}, \mathbf{K}'\mathbf{B}) \overset{c}{\approx} (\mathbf{B}, \mathbf{V})$$

where $\mathbf{K}, \mathbf{K}' \leftarrow R^{m\times m}$, $\mathbf{A}, \mathbf{U} \leftarrow R^{Qm\times m}$, and $\mathbf{B}, \mathbf{V} \leftarrow R^{m\times Qm}$.

Lemma A.1. *Let R be a finite ring, and let m and n be integers such that $m = 2n$ and $n > 3\log|R|$. Assume that $\mathbf{v} \leftarrow R^m$, and $\mathbf{v}^* \in R^m$ is sampled uniformly conditioned on $\mathbf{v}^t\mathbf{v}^* = 0_R$. If $\mathbf{a} \leftarrow R^n$, $\mathbf{b} \leftarrow R^n$, 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, note that by applying Lemma 3.10, we know that we can write $(\mathbf{v}_1^t, \mathbf{v}_2^t) \overset{s}{\approx} (\mathbf{a}^t, \mathbf{a}^t\mathbf{S})$ for some randomly sampled $\mathbf{S} \in R^{n\times n}$.

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 $\mathbf{w}_1^*$ conditioned 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 3.10 again, it follows that (with overwhelming probability) for every $\mathbf{k} \in \mathsf{RKer}(\mathbf{a})$ there exists a matrix $\mathbf{S}' \in R^{n\times n}$ such that $\mathbf{S}'\mathbf{b} = \mathbf{k}$.

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 $\mathbf{Sb}$ 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 $\mathbf{a}^t\mathbf{t} = 0_R$.

We next extend the above lemma to matrices of ring elements.

Corollary A.2. *Let R be a finite ring, and let m, n, ℓ be integers such that $m = 2n$, $n > 3\ell^2 \log |R|$, 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*

$$(\mathbf{V}, \mathbf{V}^*) \stackrel{s}{\approx} (\mathbf{W}, \mathbf{W}^*),$$

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. This lemma immediately follows by applying Lemma A.1. Note that $M_\ell(R)$ (the set of square $\ell \times \ell$ matrices over R) forms a finite ring. Therefore we can write $\mathbf{V}$, $\mathbf{V}^*$, $\mathbf{W}$, and $\mathbf{W}^*$ as "vectors" of square matrices of elements over R. These square matrices themselves form a ring, so we can view the set of matrices $\mathbf{V}$, $\mathbf{V}^*$, $\mathbf{W}$, and $\mathbf{W}^*$ as just vectors over a larger ring and apply Lemma A.1. □

We also need the following corollary, which plays an essential role in proving that "guard matrices" are distributed properly.

Corollary A.3. *Let R be a finite ring, and let m, n, ℓ be integers such that $m = 2n$, $n > 3\ell^2 \log |R|$, 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*

$$(\mathbf{V}, \mathbf{V}^*) \stackrel{s}{\approx} (\mathbf{W}, \mathbf{W}^*),$$

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. The proof follows by the applying the previous corollary to the coset that is induced by the element $\mathbf{MB}$. Recall that for the proof of the first corollary, the distribution of $\mathbf{W}_1^*$ was $\mathbf{SB} + \mathbf{K}$ where $\mathbf{K}$ is a uniformly chosen matrix from $\mathsf{RKer}(\mathbf{A})$. By changing the distribution of $\mathbf{K}$ to the uniform distribution over the following set (which is the coset that is induced by $\mathbf{MB}$)

$$\{\overline{\mathbf{K}} \mid \mathbf{A}\overline{\mathbf{K}} = \mathbf{MB}\},$$

we ensure that the equality $\mathbf{WW}^* = \mathbf{BM}$ holds. The rest of the proof is identical to that of the previous case, and hence we omit the details. (One can view the previous corollary as a special case of this one by setting $\mathbf{M}$ to be all-zero matrix.) □

We now move to lemmas that more directly relate to our guard matrices. In particular, the next two lemmas allow us to re-randomize guards in proof of the auxiliary baby subspace hiding assumption (Definition 7.6, Section 7.3).

Lemma A.4. *Let m, n, n_1 and n_2 be any integers such that*

$$m \geq 6(n_1 + n_2) \log |R|,$$

where R satisfies the requirements stated at the beginning of this section. Let the matrices $\mathbf{R}_1 \in R^{n_1 \times 2m}$, and $\mathbf{R}_2 \in R^{n_2 \times 2m}$ $\mathbf{L}_1 \in R^{2m \times n_1}$, $\mathbf{L}_2 \in R^{2m \times n_2}$, be random subject to the constraint that

$$\mathbf{R}_1\mathbf{L}_2 = \mathbf{0}_R \quad \text{and} \quad \mathbf{R}_2\mathbf{L}_1 = \mathbf{0}_R.$$

Suppose we sample sets of two matrices $\mathbf{X}_{1,i} \in R^{n \times n_1}$ and $\mathbf{X}_{2,i} \in R^{n \times n_2}$ uniformly at random for $i \in [N]$ (for some $N = \mathrm{poly}(\lambda)$; λ being the security parameter), and let $\mathbf{Z}_i \in R^{n \times 2m}$ be matrices sampled uniformly at random as well for $i \in [N]$. We claim that, given the matrices:

$$\mathbf{R}_1, \mathbf{R}_2, \mathbf{L}_1, \mathbf{L}_2$$

samples of the following form are indistinguishable:

$$(\mathbf{X}_{1,i}\mathbf{R}_1 + \mathbf{X}_{2,i}\mathbf{R}_2) \text{ and } (\mathbf{Z}_i)$$

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 $\mathbf{X}_1, \mathbf{X}_2$:

$$\begin{bmatrix} \mathbf{X}_1 & \mathbf{X}_2 \end{bmatrix} \begin{bmatrix} -- & \mathbf{R}_1 & -- \\ -- & \mathbf{R}_2 & -- \end{bmatrix}$$

Our proof will proceed by rewriting the matrices $\mathbf{R}_1, \mathbf{R}_2$ in a way that allows us to disentangle their orthogonality from $\mathbf{L}_1$ and $\mathbf{L}_2$. We will then be able to show that left multiplication by random matrices of ring elements successfully randomizes the combined matrix formed by concatenating $\mathbf{R}_1$ and $\mathbf{R}_2$ horizontally.

We will start with some rewriting. By Corollary A.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 $\widetilde{\mathbf{R}_1} \in R^{n_1 \times m}$, $\widetilde{\mathbf{R}_2} \in R^{n_2 \times m}$, $\widetilde{\mathbf{L}_1} \in R^{m \times n_1}$, $\widetilde{\mathbf{L}_2} \in R^{m \times n_2}$, $\mathbf{U} \in R^{m \times m}$, and $\mathbf{V} \in R^{m \times m}$ and set:

$$\mathbf{R}_1 = \begin{bmatrix} \widetilde{\mathbf{R}_1} & \widetilde{\mathbf{R}_1}\mathbf{U} \end{bmatrix}, \quad \mathbf{R}_2 = \begin{bmatrix} \widetilde{\mathbf{R}_1} & \widetilde{\mathbf{R}_2}\mathbf{V} \end{bmatrix}, \quad \mathbf{L}_1 = \begin{bmatrix} -\mathbf{V}\widetilde{\mathbf{L}_1} \\ \widetilde{\mathbf{L}_1} \end{bmatrix}, \quad \mathbf{L}_2 = \begin{bmatrix} -\mathbf{U}\widetilde{\mathbf{L}_2} \\ \widetilde{\mathbf{L}_2} \end{bmatrix}$$

Note that all of the product distributions between matrices are correctly preserved here. This means that, for any i, we have

$$\begin{bmatrix} \mathbf{X}_{1,i} & \mathbf{X}_{2,i} \end{bmatrix} \begin{bmatrix} -- & \mathbf{R}_1 & -- \\ -- & \mathbf{R}_2 & -- \end{bmatrix} = \begin{bmatrix} \mathbf{X}_{1,i} & \mathbf{X}_{2,i} \end{bmatrix} \begin{bmatrix} \widetilde{\mathbf{R}_1} & \mathbf{U}\widetilde{\mathbf{R}_1} \\ \widetilde{\mathbf{R}_2} & \mathbf{V}\widetilde{\mathbf{R}_2} \end{bmatrix} =$$

$$\begin{bmatrix} \mathbf{X}_1\widetilde{\mathbf{R}_1} + \mathbf{X}_2\widetilde{\mathbf{R}_2} || & \mathbf{X}_1\widetilde{\mathbf{R}_1}\mathbf{U} + \mathbf{X}_2\widetilde{\mathbf{R}_2}\mathbf{V} \end{bmatrix}$$

Then our lemma is statistically equivalent to the following: given the terms

$$\left[\widetilde{\mathbf{R}_1}, \widetilde{\mathbf{R}_1}\mathbf{U}\right], \left[\widetilde{\mathbf{R}_2}, \widetilde{\mathbf{R}_2}\mathbf{V}\right], \left[-\widetilde{\mathbf{L}_1}\mathbf{V}, \widetilde{\mathbf{L}_1}\right], \left[-\mathbf{U}\widetilde{\mathbf{L}_2}, \widetilde{\mathbf{L}_2}\right]$$

distinguish

$$\begin{bmatrix} \mathbf{X}_{1,i}\widetilde{\mathbf{R}_1} + \mathbf{X}_{2,i}\widetilde{\mathbf{R}_2} || & \mathbf{X}_{1,i}\widetilde{\mathbf{R}_1}\mathbf{U} + \mathbf{X}_{2,i}\widetilde{\mathbf{R}_2}\mathbf{V} \end{bmatrix} \text{ and } \mathbf{Z}_i$$

Let $\mathbf{C}_i \in R^{n \times m}$ and $\mathbf{D}_i \in R^{n \times m}$ be uniformly random matrices. By Theorem 3.13 (our core theorem about hardness in the output space of the RKHwPRF), we know that the following sets tuples are indistinguishable:

$$\left(\widetilde{\mathbf{R}_1}, \mathbf{X}_{1,i}\widetilde{\mathbf{R}_1}\right) \quad \text{and} \quad \left(\widetilde{\mathbf{R}_1}, \mathbf{C}_i\right)$$

$$\left(\widetilde{\mathbf{R}_2}, \mathbf{X}_{2,i}\widetilde{\mathbf{R}_2}\right) \quad \text{and} \quad \left(\widetilde{\mathbf{R}_2}, \mathbf{D}_i\right)$$

Using this indistinguishability, we can say that any adversary that can distinguish our original set of terms from random must be able to distinguish the following set of terms from random:

$$\begin{bmatrix} \mathbf{C}_i + \mathbf{D}_i & ||\mathbf{C}_i\mathbf{U} + \mathbf{D}_i\mathbf{V} \end{bmatrix} \text{ and } \mathbf{Z}_i$$

Now we can do some more rewriting. Note that

$$\begin{bmatrix}\mathbf{C}_i + \mathbf{D}_i & ||\mathbf{C}_i\mathbf{U} + \mathbf{D}_i\mathbf{V}\end{bmatrix} = \begin{bmatrix}\mathbf{C}_i + \mathbf{D}_i & ||\left(\mathbf{C}_i + \mathbf{D}_i\right)\mathbf{U} + \mathbf{D}_i\left(\mathbf{V} - \mathbf{U}\right)\end{bmatrix}$$

Now suppose we set $\mathbf{E}_i = \mathbf{C}_i + \mathbf{D}_i$ for $\mathbf{E}_i \in R^{n\times m}$. We know that the above is identically distributed to

$$\begin{bmatrix}\mathbf{E}_i & ||\mathbf{E}_i\mathbf{U} + \mathbf{D}_i\left(\mathbf{V} - \mathbf{U}\right).\end{bmatrix}$$

Since $\mathbf{C}_i$ was distributed uniformly at random and is no longer included in the proofs, we may assume that $\mathbf{E}_i$ is distributed uniformly at random.

Since $\mathbf{V}$ and $\mathbf{U}$ are uniformly random (but known), we know that, for some random (but known) $\mathbf{W} \in R^{n\times m}$, the above is identically distributed to

$$\begin{bmatrix}\mathbf{E}_i & |\mathbf{E}_i\mathbf{U} + \mathbf{D}_i\mathbf{W}\end{bmatrix}$$

Let $\mathbf{K}_i \in R^{n\times m}$ be a sequence of uniformly random matrices. By applying Theorem 3.13 again, we can see that the following are indistinguishable:

$$(\mathbf{W}, \mathbf{D}_i\mathbf{W}) \quad \text{and} \quad (\mathbf{W}, \mathbf{K}_i)$$

If we substitute this back in to our previous terms, then we have that we need to show that

$$\begin{bmatrix}\mathbf{E}_i & ||\mathbf{K}_i\end{bmatrix} \text{ and } \mathbf{Z}_i$$

are indistinguishable, which is immediately true and completes the proof. □

We now need to prove a similar lemma for the "inner" guard matrices. The proof is slightly more complicated in terms of notation, but the intuition is exactly the same.

Lemma A.5. *Let m, n_1 and n_2 be any integers such that*

$$m \geq 6(n_1 + n_2)\log|R|,$$

where R is any ring satisfying the conditions stated at the beginning of this section. Let the matrices $\mathbf{R}_{1,2} \in R^{n_2\times 2m}$, $\mathbf{L}_{2,1} \in R^{2m\times n_1}$, $\mathbf{R}_{2,1} \in R^{n_1\times 2m}$, and $\mathbf{R}_{2,2} \in R^{n_2\times 2m}$, $\mathbf{L}_{3,1} \in R^{2m\times n_1}$, and $\mathbf{L}_{3,2} \in R^{2m\times n_2}$ be random subject to the constraint that, for all relevant i:

$$\mathbf{R}_{i,1}\mathbf{L}_{i+1,2} = \mathbf{0}_R \qquad \textit{and} \qquad \mathbf{R}_{i,2}\mathbf{L}_{i+1,1} = \mathbf{0}_R.$$

Suppose we sample sets of two matrices $\mathbf{X}_{1,i} \in R^{n_1\times n_1}, \mathbf{X}_{2,i} \in R^{n_2\times n_2}$ for $i \in [N]$ (for some $N = \mathrm{poly}(\lambda)$; λ being the security parameter), and let $\mathbf{Z} \in R^{2m\times n_2}$ and $\mathbf{Z}' \in R^{n_1\times 2m}$ be sampled uniformly at random as well. We claim the following: given the terms

$$\mathbf{R}_{1,2}, \mathbf{L}_{2,1}, \mathbf{R}_{2,1}, \mathbf{R}_{2,2}, \mathbf{L}_{3,1}, \mathbf{L}_{3,2}, \mathbf{Z}, \mathbf{Z}'$$

the following sets of tuples are indistinguishable:

$$(\mathbf{L}_{2,1}\mathbf{X}_{1,i}\mathbf{R}_{2,1} + \mathbf{Z}\mathbf{X}_{2,i}\mathbf{R}_{2,2}) \quad \textit{and} \quad \left(\mathbf{L}_{2,1}\mathbf{X}_{1,i}\mathbf{Z}' + \mathbf{Z}\mathbf{X}_{2,i}\mathbf{R}_{2,2}\right)$$

Proof. Observe that, given our initial set of guard matrices (notably including $\mathbf{L}_{3,1}$ and $\mathbf{L}_{3,2}$), essentially what we are trying to prove is that sets of the following two products are indistinguishable from each other:

$$\begin{bmatrix} | & | \\ \mathbf{L}_{2,1} & \mathbf{Z} \\ | & | \end{bmatrix} \begin{bmatrix} \mathbf{X}_{1,i} & \mathbf{0}_R \\ \mathbf{0}_R & \mathbf{X}_{2,i} \end{bmatrix} \begin{bmatrix} -- & \mathbf{R}_{2,1} & -- \\ -- & \mathbf{R}_{2,2} & -- \end{bmatrix}$$

and

$$\begin{bmatrix} | & | \\ \mathbf{L}_{2,1} & \mathbf{Z} \\ | & | \end{bmatrix} \begin{bmatrix} \mathbf{X}_{1,i} & \mathbf{0}_R \\ \mathbf{0}_R & \mathbf{X}_{2,i} \end{bmatrix} \begin{bmatrix} -- & \mathbf{Z}' & -- \\ -- & \mathbf{R}_{2,2} & -- \end{bmatrix}$$

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.

We will start by using a similar rewriting strategy as in lemma A.4. By Corollary A.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 $\widetilde{\mathbf{R}_{1,2}} \in R^{n_2\times m}$, $\widetilde{\mathbf{L}_{2,1}} \in R^{m\times n_1}$, $\widetilde{\mathbf{R}_{2,1}} \in R^{n_1\times m}$, $\widetilde{\mathbf{L}_{3,2}} \in R^{m\times n_2}$, $\widetilde{\mathbf{R}_{2,2}} \in R^{n_2\times m}$, $\widetilde{\mathbf{L}_{3,1}} \in R^{m\times n_1}$, $\mathbf{V}_1 \in R^{m\times m}$, $\mathbf{U}_2 \in R^{m\times m}$, and $\mathbf{V}_2 \in R^{m\times m}$ be distributed uniformly at random. We can define our original matrices in the following way:

$$\mathbf{R}_{1,2} = \begin{bmatrix}\widetilde{\mathbf{R}_{1,2}} & \widetilde{\mathbf{R}_{1,2}}\mathbf{V}_1\end{bmatrix}, \quad \mathbf{L}_{2,1} = \begin{bmatrix}-\widetilde{\mathbf{V}_1\mathbf{L}_{2,1}} \\ \widetilde{\mathbf{L}_{2,1}}\end{bmatrix}, \quad \mathbf{R}_{2,1} = \begin{bmatrix}\widetilde{\mathbf{R}_{2,1}} & \widetilde{\mathbf{R}_{2,1}}\mathbf{U}_2\end{bmatrix}, \quad \mathbf{L}_{3,2} = \begin{bmatrix}-\mathbf{U}_2\widetilde{\mathbf{L}_{3,2}} \\ \widetilde{\mathbf{L}_{3,2}}\end{bmatrix},$$

$$\mathbf{R}_{2,2} = \begin{bmatrix}\widetilde{\mathbf{R}_{2,2}} & \widetilde{\mathbf{R}_{2,2}}\mathbf{V}_2\end{bmatrix}, \quad \mathbf{L}_{3,1} = \begin{bmatrix}-\mathbf{V}_2\widetilde{\mathbf{L}_{3,1}} \\ \widetilde{\mathbf{L}_{3,1}}\end{bmatrix}$$

Note that any adversary that can distinguish the terms as stated in the lemma from random can also distinguish the terms from random when given them in the form above (because it is more information and distributed in a statistically identical manner). So, we will assume from now on that the adversary has access to all of the relevant $\widetilde{\mathbf{R}_{i,j}}$, $\widetilde{\mathbf{L}_{i,j}}$, $\mathbf{V}_1$, $\mathbf{V}_2$, and $\mathbf{U}_2$.

As a further simplification, we set $\mathbf{Z}^T = \left[\mathbf{Z}_1^T || \mathbf{Z}_2^T\right]^T$ and $\mathbf{Z}' = [\mathbf{Z}_1' || \mathbf{Z}_2']$ for $\mathbf{Z}_1, \mathbf{Z}_2 \in R^{m\times n_1}$ and $\mathbf{Z}_1', \mathbf{Z}_2' \in R^{n_1\times m}$. We can rewrite

$$\begin{bmatrix} | & | \\ \mathbf{L}_{2,1} & \mathbf{Z} \\ | & | \end{bmatrix} \begin{bmatrix}\mathbf{X}_{1,i} & \mathbf{0}_R \\ \mathbf{0}_R & \mathbf{X}_{2,i}\end{bmatrix} \begin{bmatrix}-- & \mathbf{R}_{2,1} & -- \\ -- & \mathbf{R}_{2,2} & --\end{bmatrix}$$

as

$$\begin{bmatrix} | & | \\ \mathbf{V}_1\widetilde{\mathbf{L}_{2,1}} & \mathbf{Z}_1 \\ | & | \\ | & | \\ \widetilde{\mathbf{L}_{2,1}} & \mathbf{Z}_2 \\ | & | \end{bmatrix} \begin{bmatrix}\mathbf{X}_{1,i} & \mathbf{0}_R \\ \mathbf{0}_R & \mathbf{X}_{2,i}\end{bmatrix} \begin{bmatrix}-- & \widetilde{\mathbf{R}_{2,1}} & -- & -- & \widetilde{\mathbf{R}_{2,1}}\mathbf{U}_2 & -- \\ -- & \widetilde{\mathbf{R}_{2,2}} & -- & -- & \widetilde{\mathbf{R}_{2,2}}\mathbf{V}_2 & --\end{bmatrix}$$

$$= \begin{bmatrix} | & | \\ \mathbf{V}_1\widetilde{\mathbf{L}_{2,1}}\mathbf{X}_{1,i} & \mathbf{Z}_1\mathbf{X}_{2,i} \\ | & | \\ | & | \\ \widetilde{\mathbf{L}_{2,1}}\mathbf{X}_{1,i} & \mathbf{Z}_2\mathbf{X}_{2,i} \\ | & | \end{bmatrix} \begin{bmatrix}-- & \widetilde{\mathbf{R}_{2,1}} & -- & -- & \widetilde{\mathbf{R}_{2,1}}\mathbf{U}_2 & -- \\ -- & \widetilde{\mathbf{R}_{2,2}} & -- & -- & \widetilde{\mathbf{R}_{2,2}}\mathbf{V}_2 & --\end{bmatrix}$$

and also

$$\begin{bmatrix} | & | \\ \mathbf{L}_{2,1} & \mathbf{Z} \\ | & | \end{bmatrix} \begin{bmatrix} \mathbf{X}_{1,i} & \mathbf{0}_R \\ \mathbf{0}_R & \mathbf{X}_{2,i} \end{bmatrix} \begin{bmatrix} -- & \mathbf{Z}' & -- \\ -- & \mathbf{R}_{2,2} & -- \end{bmatrix}$$

as

$$\begin{bmatrix} | & | \\ \mathbf{V}_1\widetilde{\mathbf{L}_{2,1}} & \mathbf{Z}_1 \\ | & | \\ | & | \\ \widetilde{\mathbf{L}_{2,1}} & \mathbf{Z}_2 \\ | & | \end{bmatrix} \begin{bmatrix} \mathbf{X}_{1,i} & \mathbf{0}_R \\ \mathbf{0}_R & \mathbf{X}_{2,i} \end{bmatrix} \begin{bmatrix} -- & \mathbf{Z}'_1 & -- & -- & \mathbf{Z}'_2 & -- \\ -- & \widetilde{\mathbf{R}_{2,2}} & -- & -- & \widetilde{\mathbf{R}_{2,2}}\mathbf{V}_2 & -- \end{bmatrix}$$

$$= \begin{bmatrix} | & | \\ \mathbf{V}_1\widetilde{\mathbf{L}_{2,1}}\mathbf{X}_{1,i} & \mathbf{Z}_1\mathbf{X}_{2,i} \\ | & | \\ | & | \\ \widetilde{\mathbf{L}_{2,1}}\mathbf{X}_{1,i} & \mathbf{Z}_2\mathbf{X}_{2,i} \\ | & | \end{bmatrix} \begin{bmatrix} -- & \mathbf{Z}'_1 & -- & -- & \mathbf{Z}'_2 & -- \\ -- & \widetilde{\mathbf{R}_{2,2}} & -- & -- & \widetilde{\mathbf{R}_{2,2}}\mathbf{V}_2 & -- \end{bmatrix}$$

Since $\widetilde{\mathbf{R}_{2,1}}$ is distributed uniformly at random given all of the other terms, we can apply Theorem 3.13 and argue that

$$\begin{bmatrix} -- & \mathbf{Z}'_1 & -- & -- & \mathbf{Z}'_2 & -- \end{bmatrix} \approx \begin{bmatrix} -- & \widetilde{\mathbf{R}_{2,1}} & -- & -- & \widetilde{\mathbf{R}_{2,1}}\widetilde{\mathbf{Z}} & -- \end{bmatrix}$$

for some random $\widetilde{\mathbf{Z}} \in R^{n_1 \times n_1}$. So, we may assume we are given a term of the form

$$\begin{bmatrix} | & | \\ \mathbf{V}_1\widetilde{\mathbf{L}_{2,1}}\mathbf{X}_{1,i} & \mathbf{Z}_1\mathbf{X}_{2,i} \\ | & | \\ | & | \\ \widetilde{\mathbf{L}_{2,1}}\mathbf{X}_{1,i} & \mathbf{Z}_2\mathbf{X}_{2,i} \\ | & | \end{bmatrix} \begin{bmatrix} -- & \widetilde{\mathbf{R}_{2,1}} & -- & -- & \widetilde{\mathbf{R}_{2,1}}\mathbf{A} & -- \\ -- & \widetilde{\mathbf{R}_{2,2}} & -- & -- & \widetilde{\mathbf{R}_{2,2}}\mathbf{V}_2 & -- \end{bmatrix}$$

and must distinguish whether $\mathbf{A}$ is either $\mathbf{U}_2$ or a uniformly random $\widetilde{\mathbf{Z}}$.

Let $\mathbf{C}_{1,i} \in R^{m \times n_1}$, $\mathbf{C}_{2,i} \in R^{m \times n_2}$, and $\mathbf{C}_{3,i} \in R^{m \times n_2}$ be uniformly random matrices. By Theorem 3.13, we know that sets of the following are indistinguishable:

$$\left(\widetilde{\mathbf{L}_{2,1}}, \widetilde{\mathbf{L}_{2,1}}\mathbf{X}_{1,i}\right) \quad \text{and} \quad \left(\widetilde{\mathbf{L}_{2,1}}, \mathbf{C}_{1,i}\right)$$

$$(\mathbf{Z}, \mathbf{Z}\mathbf{X}_{2,i}) \quad \text{and} \quad \left(\mathbf{Z}, \left[\mathbf{C}_{2,i}^T || \mathbf{C}_{3,i}^T\right]^T\right)$$

Since $\mathbf{X}_{1,i}$ and $\mathbf{X}_{2,i}$ do not appear in any terms outside of ones in the form above, we can rewrite the above product as

$$\begin{bmatrix} | & | \\ \mathbf{V}_1\mathbf{C}_{1,i} & \mathbf{C}_{2,i} \\ | & | \\ | & | \\ \mathbf{C}_{1,i} & \mathbf{C}_{3,i} \\ | & | \end{bmatrix} \begin{bmatrix} -- & \widetilde{\mathbf{R}_{2,1}} & -- & -- & \widetilde{\mathbf{R}_{2,1}}\mathbf{A} & -- \\ -- & \widetilde{\mathbf{R}_{2,2}} & -- & -- & \widetilde{\mathbf{R}_{2,2}}\mathbf{V}_2 & -- \end{bmatrix}$$

If we expand this matrix, we get

$$\begin{bmatrix} \mathbf{V}_1\mathbf{C}_{1,i}\widetilde{\mathbf{R}_{2,1}} + \mathbf{C}_{2,i}\widetilde{\mathbf{R}_{2,2}} & \mathbf{V}_1\mathbf{C}_{1,i}\widetilde{\mathbf{R}_{2,1}}\mathbf{A} + \mathbf{C}_{2,i}\widetilde{\mathbf{R}_{2,2}}\mathbf{V}_2 \\ \mathbf{C}_{1,i}\widetilde{\mathbf{R}_{2,1}} + \mathbf{C}_{3,i}\widetilde{\mathbf{R}_{2,2}} & \mathbf{C}_{1,i}\widetilde{\mathbf{R}_{2,1}}\mathbf{A} + \mathbf{C}_{3,i}\widetilde{\mathbf{R}_{2,2}}\mathbf{V}_2 \end{bmatrix}$$

If we can show that it is impossible for an adversary to distinguish the form of $\mathbf{A}$ in the sets of the above tuples when given all of the other relevant terms, then we will be done. First, recall that adversary knows $\mathbf{V}_1, \mathbf{V}_2$. Additionally, recall that (square) matrices of the form

$$\begin{bmatrix} \mathbf{I} & \mathbf{X} \\ \mathbf{0}_R & \mathbf{I} \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} \mathbf{Y} & \mathbf{I} \\ \mathbf{I} & \mathbf{0}_R \end{bmatrix}$$

are invertible for any ring and for any $\mathbf{X}, \mathbf{Y}$. Now let's consider the product

$$\begin{bmatrix} \mathbf{I} & -\mathbf{V}_1 \\ \mathbf{0}_R & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{V}_1\mathbf{C}_{1,i}\widetilde{\mathbf{R}_{2,1}} + \mathbf{C}_{2,i}\widetilde{\mathbf{R}_{2,2}} & \mathbf{V}_1\mathbf{C}_{1,i}\widetilde{\mathbf{R}_{2,1}}\mathbf{A} + \mathbf{C}_{2,i}\widetilde{\mathbf{R}_{2,2}}\mathbf{V}_2 \\ \mathbf{C}_{1,i}\widetilde{\mathbf{R}_{2,1}} + \mathbf{C}_{3,i}\widetilde{\mathbf{R}_{2,2}} & \mathbf{C}_{1,i}\widetilde{\mathbf{R}_{2,1}}\mathbf{A} + \mathbf{C}_{3,i}\widetilde{\mathbf{R}_{2,2}}\mathbf{V}_2 \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{0}_R \\ -\mathbf{V}_2 & \mathbf{I} \end{bmatrix}$$

If we explicitly work out this computation, we find that

$$\begin{bmatrix} \mathbf{I} & -\mathbf{V}_1 \\ \mathbf{0}_R & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{V}_1\mathbf{C}_{1,i}\widetilde{\mathbf{R}_{2,1}} + \mathbf{C}_{2,i}\widetilde{\mathbf{R}_{2,2}} & \mathbf{V}_1\mathbf{C}_{1,i}\widetilde{\mathbf{R}_{2,1}}\mathbf{A} + \mathbf{C}_{2,i}\widetilde{\mathbf{R}_{2,2}}\mathbf{V}_2 \\ \mathbf{C}_{1,i}\widetilde{\mathbf{R}_{2,1}} + \mathbf{C}_{3,i}\widetilde{\mathbf{R}_{2,2}} & \mathbf{C}_{1,i}\widetilde{\mathbf{R}_{2,1}}\mathbf{A} + \mathbf{C}_{3,i}\widetilde{\mathbf{R}_{2,2}}\mathbf{V}_2 \end{bmatrix} =$$

$$\begin{bmatrix} \mathbf{C}_{2,i}\widetilde{\mathbf{R}_{2,2}} - \mathbf{V}_1\mathbf{C}_{3,i}\widetilde{\mathbf{R}_{2,2}} & \mathbf{C}_{2,i}\widetilde{\mathbf{R}_{2,2}}\mathbf{V}_2 - \mathbf{V}_1\mathbf{C}_{3,i}\widetilde{\mathbf{R}_{2,2}}\mathbf{V}_2 \\ \mathbf{C}_{1,i}\widetilde{\mathbf{R}_{2,1}} + \mathbf{C}_{3,i}\widetilde{\mathbf{R}_{2,2}} & \mathbf{C}_{1,i}\widetilde{\mathbf{R}_{2,1}}\mathbf{A} + \mathbf{C}_{3,i}\widetilde{\mathbf{R}_{2,2}}\mathbf{V}_2 \end{bmatrix}$$

and also that

$$\begin{bmatrix} \mathbf{C}_{2,i}\widetilde{\mathbf{R}_{2,2}} - \mathbf{V}_1\mathbf{C}_{3,i}\widetilde{\mathbf{R}_{2,2}} & \mathbf{C}_{2,i}\widetilde{\mathbf{R}_{2,2}}\mathbf{V}_2 - \mathbf{V}_1\mathbf{C}_{3,i}\widetilde{\mathbf{R}_{2,2}}\mathbf{V}_2 \\ \mathbf{C}_{1,i}\widetilde{\mathbf{R}_{2,1}} + \mathbf{C}_{3,i}\widetilde{\mathbf{R}_{2,2}} & \mathbf{C}_{1,i}\widetilde{\mathbf{R}_{2,1}}\mathbf{A} + \mathbf{C}_{3,i}\widetilde{\mathbf{R}_{2,2}}\mathbf{V}_2 \end{bmatrix} \begin{bmatrix} -\mathbf{V}_2 & \mathbf{I} \\ \mathbf{I} & \mathbf{0}_R \end{bmatrix}$$

$$= \begin{bmatrix} \mathbf{0}_R & \mathbf{C}_{2,i}\widetilde{\mathbf{R}_{2,2}} - \mathbf{V}_1\mathbf{C}_{3,i}\widetilde{\mathbf{R}_{2,2}} \\ \mathbf{C}_{1,i}\widetilde{\mathbf{R}_{2,1}}\mathbf{A} + \mathbf{C}_{3,i}\widetilde{\mathbf{R}_{2,2}}\mathbf{V}_2 & \mathbf{C}_{1,i}\widetilde{\mathbf{R}_{2,1}} + \mathbf{C}_{3,i}\widetilde{\mathbf{R}_{2,2}} \end{bmatrix}$$

This substantially simplifies our terms so we can apply our core lemmas again (and later, we can "un-cancel" the matrices we used to do this simplification since they are invertible). Let $\mathbf{E}_{1,i} \in R^{m \times m}$, $\mathbf{E}_{2,i} \in R^{m \times m}$, and $\mathbf{E}_{3,i} \in R^{m \times m}$ be uniformly random matrices. By applying Theorem 3.13 repeatedly in a blockwise manner, we know that the following are indistinguishable from random (since the $\mathbf{C}_{j,i}$ matrices appear nowhere else but these terms):

$$\left(\widetilde{\mathbf{R}_{2,1}}, \mathbf{C}_{1,i}\widetilde{\mathbf{R}_{2,1}}\right) \quad \text{and} \quad \left(\widetilde{\mathbf{R}_{2,1}}, \mathbf{E}_{1,i}\right)$$

$$\left(\widetilde{\mathbf{R}_{2,2}}, \mathbf{C}_{2,i}\widetilde{\mathbf{R}_{2,2}}\right) \quad \text{and} \quad \left(\widetilde{\mathbf{R}_{2,2}}, \mathbf{E}_{2,i}\right)$$

$$\left(\widetilde{\mathbf{R}_{2,2}}, \mathbf{C}_{3,i}\widetilde{\mathbf{R}_{2,2}}\right) \quad \text{and} \quad \left(\widetilde{\mathbf{R}_{2,2}}, \mathbf{E}_{3,i}\right)$$

This allows us to argue that the previous matrix we were examining is indistinguishable from

$$\begin{bmatrix} \mathbf{0}_R & \mathbf{E}_{2,i} - \mathbf{V}_1\mathbf{E}_{3,i} \\ \mathbf{E}_{1,i}\mathbf{A} + \mathbf{E}_{3,i}\mathbf{V}_2 & \mathbf{E}_{1,i} + \mathbf{E}_{3,i} \end{bmatrix}$$

Note that $\mathbf{E}_{2,i} - \mathbf{V}_1\mathbf{E}_{3,i}$ is distributed randomly given the rest of the terms since $\mathbf{E}_{2,i}$ is random and is not present in any of the other terms. Let $\mathbf{G}_{1,i}, \mathbf{G}_{2,i} \in R^{m \times m}$ be uniformly random matrices. We will next show that the tuples

$$(\mathbf{A}, \mathbf{V}_2, \mathbf{E}_{1,i}\mathbf{A} + \mathbf{E}_{3,i}\mathbf{V}_2, \mathbf{E}_{1,i} + \mathbf{E}_{3,i}) \quad \text{and} \, (\mathbf{A}, \mathbf{V}_2, \mathbf{G}_{1,i}, \mathbf{G}_{2,i})$$

are indistinguishable. Suppose we write $\mathbf{F}_i = \mathbf{E}_1 + \mathbf{E}_3$ and then rewrite our left tuple above as

$$(\mathbf{A}, \mathbf{V}_2, \mathbf{E}_1 (\mathbf{A} - \mathbf{V}_2) + \mathbf{F}_i\mathbf{V}_2, \mathbf{F})$$

Since $\mathbf{F}_i$ is uniformly random even given all of the other variables (since we have eliminated $\mathbf{E}_{3,i}$), we can apply a simple substitution and Theorem 3.13 to finish the indistinguishability.

Now let's finish the lemma. At this point, we have shown that, given all of the relevant matrices in the assumption statement, terms of the form

$$\begin{bmatrix} \mathbf{V}_1\mathbf{C}_{1,i}\widetilde{\mathbf{R}_{2,1}} + \mathbf{C}_{2,i}\widetilde{\mathbf{R}_{2,2}} & \mathbf{V}_1\mathbf{C}_{1,i}\widetilde{\mathbf{R}_{2,1}}\mathbf{A} + \mathbf{C}_{2,i}\widetilde{\mathbf{R}_{2,2}}\mathbf{V}_2 \\ \mathbf{C}_{1,i}\widetilde{\mathbf{R}_{2,1}} + \mathbf{C}_{3,i}\widetilde{\mathbf{R}_{2,2}} & \mathbf{C}_{1,i}\widetilde{\mathbf{R}_{2,1}}\mathbf{A} + \mathbf{C}_{3,i}\widetilde{\mathbf{R}_{2,2}}\mathbf{V}_2 \end{bmatrix}$$

are indistinguishable from those of the form

$$\begin{bmatrix} \mathbf{I} & \mathbf{V}_1 \\ \mathbf{0}_R & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{0}_R & \mathbf{H}_i \\ \mathbf{G}_{1,i} & \mathbf{G}_{2,i} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{0}_R \\ \mathbf{V}_2 & \mathbf{I} \end{bmatrix}$$

for uniformly random $\mathbf{H}_i$, $\mathbf{G}_{1,i}$, and $\mathbf{G}_{2,i}$. Since these are independent of $\mathbf{A}$, we have completed the proof. □

We next state two corollaries which are essentially just the transposes of lemmas A.4 and A.5. We are just swapping the roles of the left and right guards (which changes nothing, because they are distributed in a way that is indistinguishable) which allows us to work "backwards" in the proof of the baby subspace assumption. The proofs are immediate from the proofs of lemmas A.4 and A.5 themselves.

Corollary A.6. *Let m, n, n_1 and n_2 be any integers such that*

$$m \geq 6(n_1 + n_2)\log|R|,$$

where R is any ring satisfying the conditions stated at the beginning of this section. Let the matrices $\mathbf{R}_1 \in R^{n_1 \times 2m}$, and $\mathbf{R}_2 \in R^{n_2 \times 2m}$ $\mathbf{L}_1 \in R^{2m \times n_1}$, $\mathbf{L}_2 \in R^{2m \times n_2}$, be random subject to the constraint that

$$\mathbf{R}_1\mathbf{L}_2 = \mathbf{0}_R \quad \text{and} \quad \mathbf{R}_2\mathbf{L}_1 = \mathbf{0}_R.$$

Suppose we sample sets of two matrices $\mathbf{X}_{1,i} \in R^{n_1 \times n}, \mathbf{X}_{2i} \in R^{n_2 \times n}$ uniformly at random for $i \in [N]$ (for some $N = \text{poly}(\lambda)$; λ being the security parameter), and let $\mathbf{Z}_i \in R^{2m \times n}$ be matrices sampled uniformly at random as well for $i \in [N]$. We claim that, given the matrices:

$$\mathbf{R}_1, \mathbf{R}_2, \mathbf{L}_1, \mathbf{L}_2$$

samples of the following form are indistinguishable:

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

Corollary A.7. *Let m, n_1 and n_2 be any integers such that*

$$m \geq 6(n_1 + n_2)\log|R|,$$

where R is any ring satisfying the conditions stated at the beginning of this section. Let the matrices $\mathbf{R}_{1,2} \in R^{n_2 \times 2m}$, $\mathbf{L}_{2,1} \in R^{2m \times n_1}$, $\mathbf{R}_{2,1} \in R^{n_1 \times 2m}$, and $\mathbf{R}_{2,2} \in R^{n_2 \times 2m}$, $\mathbf{L}_{3,1} \in R^{2m \times n_1}$, and $\mathbf{L}_{3,2} \in R^{2m \times n_2}$ be random subject to the constraint that, for all relevant i:

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

Suppose we sample sets of two matrices $\mathbf{X}_{1,i} \in R^{n_1 \times n_1}, \mathbf{X}_{2,i} \in R^{n_2 \times n_2}$ for $i \in [N]$ (for some $N = \text{poly}(\lambda)$; λ being the security parameter), and let $\mathbf{Z}' \in R^{2m \times n_1}$ and $\mathbf{Z} \in R^{n_2 \times 2m}$ be sampled uniformly at random as well. We claim the following: given the terms

$$\mathbf{R}_{1,2}, \mathbf{R}_{2,2}\mathbf{L}_{2,1}, \mathbf{R}_{2,1}, \mathbf{R}_{2,2}, \mathbf{L}_{3,2}, \mathbf{Z}, \mathbf{Z}'$$

the following sets of tuples are indistinguishable:

$$(\mathbf{L}_{2,1}\mathbf{X}_{1,i}\mathbf{R}_{2,1} + \mathbf{L}_{2,2}\mathbf{X}_{2,i}\mathbf{Z}) \quad \text{and} \quad \left(\mathbf{Z}'\mathbf{X}_{1,i}\mathbf{R}_{2,1} + \mathbf{L}_{2,2}\mathbf{X}_{2,i}\mathbf{Z}\right)$$