GGH15 Beyond Permutation Branching Programs: Proofs, Attacks, and Candidates

GGH15 Beyond Permutation Branching Programs: Proofs, Attacks, and Candidates

Yilei Chen
Boston University*

Vinod Vaikuntanathan
MIT CSAIL†

Hoeteck Wee
CNRS and ENS‡

April 17, 2018

Abstract

We carry out a systematic study of the GGH15 graded encoding scheme used with *general* branching programs. This is motivated by the fact that general branching programs are more efficient than permutation branching programs and also substantially more expressive in the read-once setting. Our main results are as follows:

- **Proofs.** We present new constructions of private constrained PRFs and lockable obfuscation, for constraints (resp. functions to be obfuscated) that are computable by general branching programs. Our constructions are secure under LWE with subexponential approximation factors. Previous constructions of this kind crucially rely on the permutation structure of the underlying branching programs. Using general branching programs allows us to obtain more efficient constructions for certain classes of constraints (resp. functions), while posing new challenges in the proof, which we overcome using new proof techniques.
- **Attacks.** We extend the previous attacks on indistinguishability obfuscation (iO) candidates that use GGH15 encodings. The new attack simply uses the rank of a matrix as the distinguisher, so we call it a "rank attack". The rank attack breaks, among others, the iO candidate for general read-once branching programs by Halevi, Halevi, Shoup and Stephens-Davidowitz (CCS 2017).
- **Candidate Witness Encryption and iO.** Drawing upon insights from our proofs and attacks, we present simple candidates for witness encryption and iO that resist the existing attacks, using GGH15 encodings. Our candidate for witness encryption crucially exploits the fact that formulas in conjunctive normal form (CNFs) can be represented by general, *read-once* branching programs.

*E-mail: cheny1@bu.edu. Supported by the NSF MACS project. Part of this work was done while visiting ENS.

†E-mail: vinodv@csail.mit.edu. Research supported in part by NSF Grants CNS-1350619 and CNS-1414119, Alfred P. Sloan Research Fellowship, Microsoft Faculty Fellowship, the NEC Corporation and a Steven and Renee Finn Career Development Chair from MIT. This work was also sponsored in part by the Defense Advanced Research Projects Agency (DARPA) and the U.S. Army Research Office under contracts W911NF-15-C-0226 and W911NF-15-C-0236.

‡E-mail: wee@di.ens.fr. Research supported by ERC Project aSCEND (H2020 639554). Part of this work was done while visiting CQT.

Contents

1 Introduction

Graph-induced graded encodings – henceforth called GGH15 encodings – were put forth by Gentry, Gorbunov and Halevi [GGH15] as a candidate instantiation of (approximate) cryptographic multilinear maps [BS03, GGH13a], with the hope that these encodings could in turn be used to build advanced cryptographic primitives whose security is related to the hardness of the learning with errors (LWE) problem [Reg05]. In addition, following [GGH13a, GGH+13b], the same work presented candidate constructions of multi-party key exchange and indistinguishability obfuscation (iO) starting from these graded encoding schemes.

In the last few years, a very fruitful line of works has shed a great deal of insight into the use of GGH15 encodings in two complementary settings: constructing security reductions from LWE (partially validating the intuition in GGH15), and demonstrating efficient attacks. The former include constructions of private constrained pseudorandom functions (PRFs) [CC17], lockable obfuscation (aka obfuscating the "compute-then-compare" functionality) [GKW17a, WZ17] and encryption schemes that constitute counter-examples for circular security [KW16, GKW17b]. The latter include efficient attacks [CLLT16, CGH17] on the key exchange and iO candidates described in [GGH15]. One of the key distinctions between the two settings is whether an adversary can obtain encodings of zero from honest evaluations. For all the applications that can be based on LWE, the adversary cannot trivially obtain encodings of zero; whereas the attacks apply only to settings where the adversary can trivially obtain encodings of zero. There is much grey area in between, where we neither know how to obtain encodings of zero nor are we able to prove security based on LWE (e.g., in the setting of witness encryption).

This work. In this work, we explore the use of GGH15 encodings together with general (non-permutation) matrix branching programs. In particular, we present (i) new constructions of private constrained PRFs and lockable obfuscation from LWE, (ii) new attacks on iO candidates, and (iii) new candidates for iO and witness encryption that resist our new attacks as well as prior attacks. At the core of these results are new techniques and insights into the use of GGH15 encodings for a larger class of branching programs.

Most of the prior constructions and candidates for the primitives we consider follow the template laid out in [GGH+13b]: start with the class of NC^1 circuits, represented using permutation branching programs, which are specified by a collection of permutation matrices $\{\mathbf{M}_{i,b}\}_{i\in[h],b\in\{0,1\}}$. Computation in such a program proceeds by taking a subset product of these matrices, where the choice of the subset is dictated by the input but the order in which the matrices are multiplied is oblivious to the input. To cryptographically "protect" this computation, we will first pre-process and randomize $\{\mathbf{M}_{i,b}\}$ to obtain a new collection of matrices $\{\hat{\mathbf{S}}_{i,b}\}$, and then encode the latter using graded encodings. Functionality (e.g. evaluation in lockable obfuscation and iO) relies on the fact that we can check whether some subset product of the $\hat{\mathbf{S}}_{i,b}$'s is zero (or the identity matrix) using the underlying graded encodings. Any security proof or attack would of course depend on the class of matrices $\mathbf{M}_{i,b}$'s we start out with, and how the $\hat{\mathbf{S}}_{i,b}$'s are derived.

Beyond permutation matrices. From a feasibility point of view, working with permutation matrices is without loss of generality. We know that any NC^1 circuit (or even a logspace computation) can be represented as a permutation matrix branching program [Bar86]. Moreover, any general branching program, where the underlying matrices are possibly low-rank, can be converted to a

permutation branching program with a polynomial blow-up in the number and dimensions of these matrices. Nonetheless, there are advantages to working with more general, not necessarily permutation or full-rank, branching programs:

- The first is concrete efficiency. For instance, representing equality or point functions on ℓ-bit string would use $O(\ell^2)$ constant-width matrices with permutation branching programs, but just 2ℓ width-one matrices (i.e. entries) with general branching programs.

- The second is that in the read-once setting, general branching programs are more expressive than permutation branching programs. The restriction to read-once branching programs is useful in applications such as iO and witness encryption, as they allow us to disregard "multiplicative bundling" factors that protect against mixed-input attacks, which in turn yields much more efficient constructions. This was shown in a recent work of Halevi, Halevi, Shoup and Stephens-Davidowitz (HHSS) [HHSS17], which presented an iO candidate for read-once branching programs based on GGH15 encodings. Their candidate is designed for general read-once branching programs, as read-once permutation branching programs only capture an extremely limited class of functions.

This raises the natural question of the security of GGH15-based constructions when applied to general (non-permutation, possibly low-rank) matrix branching programs, as is exactly the focus of this work. Indeed, the afore-mentioned proof techniques *and* attacks break down in this setting. In particular, the HHSS iO candidate appears to resist the existing attacks in [CLLT16, CGH17], thanks in part to the use of low-rank matrices (cf. [HHSS17, Section 1.2]).

We proceed to describe our results and techniques in more detail.

1.1 Our Results I: New Cryptographic Constructions from LWE

We present new constructions of private constrained PRFs and lockable obfuscation that work directly with general matrix branching programs. As with prior works, our constructions are secure under the LWE assumption with subexponential approximation factors. Our result generalizes the previous constructions in [CC17, GKW17a, WZ17] which only work for permutation branching programs, and yields improved concrete efficiency for several interesting classes of functions that can be represented more efficiently using general branching programs, as described next.

- Lockable obfuscation [GKW17a, WZ17] refers to the average-case secure virtual black-box (VBB) obfuscation for a class of functionalities $\mathbf{C}[f, y]$ which, on input x, output 1 if $f(x) = y$ and 0 otherwise. The average-case refers (only) to a uniformly random choice of y (more generally, y with sufficient min-entropy). For lockable obfuscation, we obtain improved constructions for a class of "compute" functions where each output bit is computed using a general branching program applied to the input x (whereas [GKW17a, WZ17] require permutation branching programs). To illustrate the efficiency gain, consider the case where each output bit of the underlying function f computes a disjunction or conjunction of the ℓ input bits. In this case, we achieve up to a quadratic gain in efficiency due to our support for general branching programs. This class generalizes the distributional conjunction obfuscator studied in [BR13, BVWW16, WZ17].

- Private puncturable PRFs are an important special case of constrained PRFs, with many applications such as 2-server private information retrieval (PIR) [BKM17]. We obtain a very

simple private puncturable PRF with a quadratic efficiency improvement over the recent GGH15-based construction of Canetti and Chen [CC17]. Nonetheless, our construction is admittedly less efficient –for most settings of parameters– than the more complex constructions in [BKM17, BTVW17] that combines techniques from both fully-homomorphic and attribute-based encryption.

Next, we provide a very brief overview of our techniques, and defer a more detailed technical overview to Section 2.

New constructions and proof techniques. A GGH15 encoding of a low-norm matrix $\hat{\mathbf{S}}$ w.r.t. two matrices $\mathbf{A}_0$ and $\mathbf{A}_1$ is defined to be along the edge $\mathbf{A}_0 \mapsto \mathbf{A}_1$ and is computed as

$$\mathbf{D} \leftarrow \mathbf{A}_0^{-1}(\hat{\mathbf{S}}\mathbf{A}_1 + \mathbf{E})$$

where for all $\mathbf{A}, \mathbf{Y}$ with proper dimensions, the notation $\mathbf{D} \leftarrow \mathbf{A}^{-1}(\mathbf{Y})$ means that $\mathbf{D}$ is a random low-norm matrix such that $\mathbf{AD} = \mathbf{Y} \bmod q$.

The constructions in [CC17, GKW17b, GKW17a, WZ17] encode any permutation matrix $\mathbf{M} \in \{0,1\}^{w \times w}$ as a GGH15 encoding of $\hat{\mathbf{S}} = \mathbf{M} \otimes \mathbf{S}$ (see example in Remark 5.5), i.e.

$$\mathbf{A}_0^{-1}((\mathbf{M} \otimes \mathbf{S})\mathbf{A}_1 + \mathbf{E})$$

for a random low-norm $\mathbf{S}$. The crux of the analysis is to show that $\mathbf{M}$ is hidden under the LWE assumption, namely: for any *permutation* matrix $\mathbf{M} \in \{0,1\}^{w \times w}$,

$$(\mathbf{A}_0, \mathbf{A}_0^{-1}((\mathbf{M} \otimes \mathbf{S})\mathbf{A}_1 + \mathbf{E})) \approx_c (\mathbf{A}_0, \mathbf{V}) \tag{1}$$

where $\mathbf{A}_0, \mathbf{A}_1$ are uniformly random over $\mathbb{Z}_q$, $\mathbf{S}, \mathbf{V}, \mathbf{E}$ are random low-norm matrices, $\approx_c$ stands for computational indistinguishable. The proof of (1) follows quite readily from the fact that given any permutation matrix $\mathbf{M} \in \{0,1\}^{w \times w}$, we have:

$$(\mathbf{A}, (\mathbf{M} \otimes \mathbf{S})\mathbf{A} + \mathbf{E}) \approx_c (\mathbf{A}, \mathbf{U})$$

under the LWE assumption, where $\mathbf{U}$ is uniformly random.

However, this statement is false for arbitrary matrices $\mathbf{M}$, take for instance $\mathbf{M} = \mathbf{0}^{w \times w}$, the all-0 matrix. Indeed, the reader can easily come up with rank-$(w-1)$ matrices $\mathbf{M}$ for which equation (1) fails to hold.

In our construction, we encode an arbitrary matrix $\mathbf{M}$ as a GGH15 encoding of

$$\hat{\mathbf{S}} = \begin{pmatrix} \mathbf{M} \otimes \mathbf{S} & \\ & \mathbf{S} \end{pmatrix}$$

That is, we append $\mathbf{S}$ along the diagonal. We then establish the following analogue of (1) under the LWE assumption: for any *arbitrary* $\mathbf{M} \in \{0,1\}^{w \times w}$,

$$\left(\mathbf{J}\mathbf{A}_0, \mathbf{A}_0^{-1}\left(\begin{pmatrix} \mathbf{M} \otimes \mathbf{S} & \\ & \mathbf{S} \end{pmatrix}\mathbf{A}_1 + \mathbf{E}\right)\right) \approx_c \left(\mathbf{J}\mathbf{A}_0, \mathbf{V}\right) \tag{2}$$

where $\mathbf{J}$ is any matrix of the form $[\star \mid \mathbf{I}]$, and $\mathbf{A}_0, \mathbf{A}_1, \mathbf{S}, \mathbf{V}, \mathbf{E}$ are distributed as in (1). This statement is qualitatively incomparable with (1): it is stronger in that it works for *arbitrary* $\mathbf{M}$, but weaker in that the distinguisher only sees partial information about $\mathbf{A}_0$.

Proving the statement in (2) requires a new proof strategy where we will treat $\mathbf{S}$ (instead of $\mathbf{A}_0, \mathbf{A}_1$) as a public matrix known to the distinguisher. In particular, we start with taking the bottom part of $\mathbf{A}_1$ as the LWE secret, in conjunction with the public $\mathbf{S}$ in the bottom-right diagonal; then use an extension of the trapdoor sampling lemma by Gentry et al. [GPV08] to produce an "oblique" (while *statistically* indistinguishable) preimage sample using only the trapdoor of the top part of $\mathbf{A}_0$; finally argue that the "oblique" sample is *computationally* indistinguishable from random Gaussian using the top part of $\mathbf{A}_0$ as the LWE secret. Walking through these steps requires new techniques on analyzing the trapdoor sampling detailed in Section 4. We refer the readers to Sections 2.2.1 and 5.3 for further explanation of the proof techniques.

Next, we show that the weaker guarantee in (2) (in that the distinguisher gets $\mathbf{J}\mathbf{A}_0$ instead of $\mathbf{A}_0$) is sufficient for constructions of constrained PRFs and lockable obfuscation based on GGH15 encodings; this yields new constructions that are directly applicable to general, non-permutation matrix branching programs.

1.2 Our Results II: New Attacks on iO Candidates

Next, we turn our attention to iO, where adversaries can obtain encodings of zero through honest evaluations. Concretely, we focus on iO candidates that follow the [GGH+13b] template described earlier in the introduction: start with a branching program $\{\mathbf{M}_{i,b}\}$, pre-process and randomize $\{\mathbf{M}_{i,b}\}$ to obtain a matrices $\{\hat{\mathbf{S}}_{i,b}\}$, and encode the latter using GGH15 encodings.

We present an attack that run in time $\mathsf{size}^{O(c)}$ for general read-c branching programs of size size. In particular, we have a polynomial-time attack when c is constant, as is the case for the iO candidate in [HHSS17] which corresponds to $c = 1$. Our attack covers various "safeguards" in the literature, such as Kilian-style randomization, multiplicative bundling, and diagonal padding. We refer the readers to Section 9.2 for a precise description of the applicability of the attacks.

Attack overview. Our attack is remarkably simple, and proceeds in two steps:

1. Compute a matrix $\mathbf{V}$ whose (i, j)'th entry correspond to an iO evaluation on input $x^{(i)} \mid y^{(j)}$ that yields an encoding of zero. The dimensions of $\mathbf{V}$ and the number of evaluations is polynomial in size^c.

2. Output the rank of $\mathbf{V}$ (over $\mathbb{Z}$). More precisely, check if $\mathsf{rank}(\mathbf{V})$ is above some threshold.

Step 1 was used in the attack of Coron et al. [CLLT16] and Chen et al. [CGH17], both originated from the zeroizing attack of Cheon et al. [CHL+15] on CLT13 [CLT13]. The novelty of our analysis lies in showing that $\mathsf{rank}(\mathbf{V})$ leaks information about the $\hat{\mathbf{S}}_{i,b}$'s and thus the plaintext branching program matrices $\mathbf{M}_{i,b}$'s. So we call the attack a "rank attack".

Our attack improves upon the previous attack of Chen et al. [CGH17] on GGH15-based iO candidates in several ways: (i) we have a classical as opposed to a quantum attack, and (ii) it is applicable to a larger class of branching programs, i.e. branching programs that are not necessarily input-partitioned or using permutation matrices.

Why the rank-attack works? To get a taste of the rank-attack, let's consider an oversimplified description of the iO candidates based on GGH15 encodings. Let $\{\hat{\mathbf{S}}_{i,b}\}$ be the randomization of

plaintext matrices $\{\mathbf{M}_{i,b}\}$. Then the obfuscated code is the GGH15 encodings of the $\hat{\mathbf{S}}_{i,b}$ matrices

$$\mathbf{A}_0, \{\mathbf{D}_{i,b}\}_{i\in[h],b\in\{0,1\}} \text{ where } \mathbf{D}_{i,b} \leftarrow \mathbf{A}_{i-1}^{-1}\left(\hat{\mathbf{S}}_{i,b}\mathbf{A}_i + \mathbf{E}_{i,b}\right)$$

Evaluation proceeds by first computing the product of $\mathbf{A}_0$ with the subset product of the $\mathbf{D}_{i,b}$ matrices. As an example, for the obfuscation of a 3-step branching program that computes all-0 functionality, the evaluation on input $x = 000$ gives

$$\mathsf{Eval}(x) = \mathbf{A}_0 \cdot \mathbf{D}_{1,0} \cdot \mathbf{D}_{2,0} \cdot \mathbf{D}_{3,0} = \hat{\mathbf{S}}_{1,0}\hat{\mathbf{S}}_{2,0}\mathbf{E}_{3,0} + \hat{\mathbf{S}}_{1,0}\mathbf{E}_{2,0}\mathbf{D}_{3,0} + \mathbf{E}_{1,0}\mathbf{D}_{2,0}\mathbf{D}_{3,0} \tag{3}$$

To give a sense of why computing the rank is useful in an attack, we make a further simplification, that suppose we manage to learn the monomial

$$\hat{\mathbf{S}}_{1,0}\mathbf{E}_{2,0}\mathbf{D}_{3,0} \in \mathbb{Z}^{t\times m}.$$

W.h.p., the Gaussians $\mathbf{E}_{2,0}, \mathbf{D}_{3,0}$ and therefore its product $\mathbf{E}_{2,0}\mathbf{D}_{3,0}$ are full rank (over $\mathbb{Z}$), so the rank of this term is that of $\hat{\mathbf{S}}_{1,0}$, which leaks some information about the rank of $\mathbf{M}_{1,0}$. Note that learning the rank of $\mathbf{M}_{1,0}$ leaks no useful information for permutation branching programs, but is sufficient to break iO for general branching programs.

In actuality, a single evaluation corresponding to an encoding of zero only provides a single value in $\mathbb{Z}$, which is a *sum* of products of the form above, multiplied by some left and right bookend vectors. To extract the important information out of the summation of random-looking terms, we will first form a matrix $\mathbf{V}$ of evaluations on appropriately chosen inputs. The matrix $\mathbf{V}$ has the property that it factors into the product of two matrices $\mathbf{V} = \mathbf{X} \cdot \mathbf{Y}$. We proceed analogously to the toy example in two steps with $\mathbf{X}, \mathbf{Y}$ playing the roles of $\hat{\mathbf{S}}_{1,0}$ and $\mathbf{E}_{2,0} \cdot \mathbf{D}_{3,0}$:

1. argue that $\mathbf{Y}$ is non-singular over $\mathbb{Q}$ so that $\mathsf{rank}(\mathbf{V}) = \mathsf{rank}(\mathbf{X})$, and
2. argue that $\mathsf{rank}(\mathbf{X})$ leaks information about the underlying branching program.

So far we have described what the analysis looks like for the read-once branching programs (i.e. $c = 1$). For the case of $c > 1$, the analysis has the flavor of converting the obfuscated code of a read-c branching program into the read-once setting, using the "tensor switching lemmas" from previous attacks [ADGM17, CLLT17] on iO candidates that use GGH13 and CLT13.

1.3 Our Results III: New Candidates

Given the insights from our proofs and attacks, we present simple candidates for witness encryption and iO based on GGH15 encodings. Our witness encryption candidate relies on the observation from [GLW14] that to build witness encryption for general NP relations, it suffices to build witness encryption for CNF formulas, and that we can represent CNF formulas using general, read-once branching programs. The ciphertext corresponding to a formula Ψ and a message $\mu \in \{0,1\}$ is of the form described in (2), namely

$$\mathbf{J}\mathbf{A}_0, \left\{\mathbf{A}_{i-1}^{-1}\left(\begin{pmatrix}\mathbf{M}_{i,b}\otimes\mathbf{S}_{i,b} & \\ & \mu\mathbf{S}_{i,b}\end{pmatrix}\mathbf{A}_i + \mathbf{E}_{i,b}\right)\right\}$$

where $\mathbf{J}$ is a specific matrix of the form $[\star \mid \mathbf{I}]$ and the $\mathbf{M}_{i,b}$'s are the read-once branching program representing Ψ.

Starting from the witness encryption candidate, we also present an iO candidate for NC^1 circuits that appear to resist our rank attack as well as all prior attacks. In order to thwart the rank attack, our iO candidate necessarily reads each input bit $\omega(1)$ times. To then prevent mixed-input attacks, we rely on an extension of multiplicative bundling factors used in prior works that uses matrices instead of scalars.

We stress that an important design goal in these candidates is simplicity so as to facilitate the security analysis. We believe and anticipate that any attacks or partial security analysis for these candidates (perhaps in some weak idealized model cf. [GMM+16]) would enhance our understanding of witness encryption and obfuscation.

1.4 Discussion and Open problems

Perspective. The proposal of candidate multilinear maps [GGH13a] from lattice-type assumptions in 2013 has triggered a major paradigm shift in cryptography and enabled numerous cryptographic applications, most notably indistinguishability obfuscation [GGH+13b]. Among the three multilinear maps candidates [GGH13a, CLT13, GGH15], GGH15 is the only one that has served as a basis for new cryptographic applications based on established lattice problems, as demonstrated in e.g. [CC17, GKW17b, GKW17a, WZ17]. We believe that extending the safe settings of GGH15 (where security can be based on the LWE assumption), as explored in this work through the generalized GGH15 framework as well as both proofs and attacks, will pave the way towards new cryptographic constructions.

Open problems. We conclude with a number of open problems:

- Study the security of our candidate for witness encryption, either prove security under instance-independent assumptions, or find a direct attack on the scheme. For the former (i.e., prove security), the only proof strategy in the existing literature is to build and prove a so-called positional witness encryption scheme [GLW14], for which the security definition allows the adversary to obtain encodings of zeroes. Unfortunately, the natural extensions of our candidate witness encryption scheme to a positional variant are susceptible to the rank attack in the presence of encodings of zeroes. For the latter (i.e., directly attack the scheme), all existing attack strategies on GGH15 encodings as used in our candidate require encodings of zeroes, which are not readily available in the witness encryption setting.

- Find a polynomial-time attack for iO candidates for branching programs where every input repeats $c = O(\lambda)$ time where λ is the security parameter. The analysis of known attacks, including our rank attack, yields running times that grow exponentially with c. There are possibilities that the analysis is not tight and the rank attack or prior attacks could in fact succeed with a smaller running time. However we have not detected such a phenomenon with experiments for small c.

- Note that all our candidate constructions are of the form: $\mathbf{A}_J, \{\mathbf{D}_{i,b}\}_{i\in[h],b\in\{0,1\}}$ and evaluation/decryption proceeds by first computing $\mathbf{A}_J\mathbf{D}_{\mathbf{x}'} := \mathbf{A}_J \prod_{i=1}^{h} \mathbf{D}_{i,x'_i}$ for some $\mathbf{x}' \in \{0,1\}^h$. Consider the following restricted class of adversaries that only gets oracle access to $\mathbf{x}' \mapsto \mathbf{A}_J\mathbf{D}_{\mathbf{x}'}$ instead of $\mathbf{A}_j, \{\mathbf{D}_{i,b}\}_{i\in[h],b\in\{0,1\}}$. Note that our rank attack as well as various mixed-input and zeroizing attacks can all be implemented using this restricted adversaries. Can we

prove (or break) security of our witness encryption or iO candidates against this restricted class of adversaries under some reasonable instance-independent assumptions?

Independent work. Variants of our new lemmas related to lattice preimage sampling in Section 4 were presented in an independent work of Goyal, Koppula and Waters [GKW18], for different purposes from ours. In [GKW18], the lemmas were used as intermediate building blocks en route a collusion resistant traitor tracing scheme based on the LWE assumption.

1.5 Reader's guide

The rest of the article is organized as follows. Section 2 provides a more detailed overview of our techniques. Section 5 gives a formal construction of the generalized-GGH15 encoding, the security notions, and the main technical proof that suffices for the applications mentioned in Sections 7 and 8. The technical proof requires new lemmas developed in Section 4. The applications require terminologies for matrix branching programs given in Section 6.

The attacks are described in Section 9. It is self-contained, so readers who are only interested in attacks can jump directly to Section 9. Sections 10 and 11 give the witness encryption and iO candidates. Their descriptions are self-contained. But to get the rationale behind the candidates the readers probably have to refer to the previous sections.

2 Technical Overview

In this section, we present a more detailed overview of our techniques. We briefly describe the notation used in this overview and the paper, and refer the reader to Section 3 for more details. We use boldface upper-case and lower-case letters for matrices and vectors respectively. Given a bit-string $\mathbf{x} \in \{0,1\}^h$, we use $\mathbf{M}_{\mathbf{x}}$ to denote matrix subset product $\prod_{i=1}^{h} \mathbf{M}_{i,x_i}$. Given matrices $\mathbf{A}, \mathbf{B}$, we use $\mathbf{A}^{-1}(\mathbf{B})$ to denote a random low-norm Gaussian $\mathbf{D}$ satisfying $\mathbf{AD} = \mathbf{B} \bmod q$. Two probability distributions are connected by $\approx_s$ or $\approx_c$ if they are statistically close or computationally indistinguishable.

2.1 Generalized GGH15 Encodings

In this work, we think of GGH15 as encoding two collections of matrices, one collection is arbitrary and the other one is random, and computing some function γ of a subset product of these matrices; we refer to this as (generalized) γ-GGH15 encodings.[1] That is, the γ-GGH15 encoding takes as input two collections of matrices $\{\mathbf{M}_{i,b}\}_{i\in[\ell],b\in\{0,1\}}$, $\{\mathbf{S}_{i,b}\}_{i\in[\ell],b\in\{0,1\}}$, an additional matrix $\mathbf{A}_\ell$, and the output is a collection of matrices

$$\mathbf{A}_0, \{\mathbf{D}_{i,b}\}_{i\in[\ell],b\in\{0,1\}}$$

such that for all $\mathbf{x} \in \{0,1\}^\ell$, we have

$$\mathbf{A}_0 \cdot \mathbf{D}_{\mathbf{x}} \approx \gamma(\mathbf{M}_{\mathbf{x}}, \mathbf{S}_{\mathbf{x}}) \cdot \mathbf{A}_\ell$$

[1] See Remark 5.2 for a comparison with the original GGH15 encodings.

where $\mathbf{M}_{\mathbf{x}}, \mathbf{D}_{\mathbf{x}}, \mathbf{S}_{\mathbf{x}}$ denotes subset-product of matrices as defined earlier. Here,

$$\mathbf{M}_{i,b} \in \{0,1\}^{w \times w}, \mathbf{S}_{i,b} \in \mathbb{Z}^{n \times n}, \mathbf{A}_0, \mathbf{A}_\ell \in \mathbb{Z}_q^{\gamma(w,n) \times m}, \mathbf{D}_{i,b} \in \mathbb{Z}^{m \times m}.$$

Intuitively, we also want to hide the $\mathbf{M}_{i,b}$'s, which we will come back to after describing the choices for γ and the construction.

Choices for γ. There are several instantiations for γ in the literature [GGH15, BVWW16, CC17, GKW17a, WZ17, GGH+13b, HHSS17]:

$$\gamma_\times(\mathbf{M}, \mathbf{S}) = \mathbf{MS}, \ \gamma_\otimes(\mathbf{M}, \mathbf{S}) := \mathbf{M} \otimes \mathbf{S}, \ \gamma_{\text{diag}}(\mathbf{M}, \mathbf{S}) := \begin{pmatrix} \mathbf{M} & \\ & \mathbf{S} \end{pmatrix}$$

where the first $\gamma_\times$ requires working with rings so that multiplication commutes. More generally, for the construction, we require that γ be multiplicatively homomorphic, so that

$$\gamma(\mathbf{M}, \mathbf{S})\gamma(\mathbf{M}', \mathbf{S}') = \gamma(\mathbf{MM}', \mathbf{SS}')$$

as is clearly satisfied by the three instantiations above.

The γ-GGH15 construction. We briefly describe the construction of γ-GGH15 encodings implicit in [GGH15], from the view-point of "cascaded cancellations" [KW16, GKW17b]. The starting point of the construction is to expand $\gamma(\mathbf{M}_{\mathbf{x}}, \mathbf{S}_{\mathbf{x}}) \cdot \mathbf{A}_\ell$ using multiplicative homomorphism as a matrix product

$$\gamma(\mathbf{M}_{\mathbf{x}}, \mathbf{S}_{\mathbf{x}}) \cdot \mathbf{A}_\ell = \prod_{i=1}^{\ell} \gamma(\mathbf{M}_{i,x_i}, \mathbf{S}_{i,x_i}) \cdot \mathbf{A}_\ell$$

Next, it randomizes the product by sampling random (wide, rectangular) matrices $\mathbf{A}_0, \ldots, \mathbf{A}_{\ell-1}$ over $\mathbb{Z}_q$ along with their trapdoors, and rewrites the product as a series of "cascaded cancellations":

$$\gamma(\mathbf{M}_{\mathbf{x}}, \mathbf{S}_{\mathbf{x}}) \cdot \mathbf{A}_\ell = \mathbf{A}_0 \cdot \prod_{i=1}^{\ell} \mathbf{A}_{i-1}^{-1}(\gamma(\mathbf{M}_{i,x_i}, \mathbf{S}_{i,x_i})\mathbf{A}_i)$$

where $\mathbf{A}_{i-1}^{-1}(\cdot)$ denotes random low-norm Gaussian pre-images as defined earlier. [2]

For functionality, it suffices to define $\mathbf{D}_{i,b}$ to be $\mathbf{A}_{i-1}^{-1}(\gamma(\mathbf{M}_{i,b}, \mathbf{S}_{i,b})\mathbf{A}_i)$, but that would not be sufficient to hide the underlying $\mathbf{M}_{i,b}$'s. Instead, the construction introduces additional error terms $\{\mathbf{E}_{i,b}\}_{i \in [\ell], b \in \{0,1\}}$, and defines[3]

$$\mathbf{D}_{i,b} \leftarrow \mathbf{A}_{i-1}^{-1}(\gamma(\mathbf{M}_{i,b}, \mathbf{S}_{i,b})\mathbf{A}_i + \mathbf{E}_{i,b})$$

[2] A reader who is familiar with Kilian's randomization for branching programs should notice the similarity. In Kilian's randomization, we randomize the product

$$\mathbf{M}_{\mathbf{x}} = \prod_{i=1}^{\ell} \mathbf{R}_{i-1}^{-1} \mathbf{M}_{i,x_i} \mathbf{R}_i$$

by picking random invertible matrices $\mathbf{R}_1, \ldots, \mathbf{R}_{\ell-1}$ along with $\mathbf{R}_0 = \mathbf{R}_\ell = \mathbf{I}$. Here, we replace the square matrices $\mathbf{R}_i$'s with wide rectangular matrices $\mathbf{A}_i$'s, and change from left-multiplying $\mathbf{R}_{i-1}^{-1}$ to sampling a random Gaussian preimage of $\mathbf{A}_{i-1}$.

[3] In the GGH15 terminology, $\mathbf{D}_{i,b}$ would be an encoding of $\gamma(\mathbf{M}_{i,b}, \mathbf{S}_{i,b})$ relative to the path $i - 1 \mapsto i$.

Observe that for all $\mathbf{x} \in \{0,1\}^\ell$, we have

$$\mathbf{A}_0 \cdot \mathbf{D}_\mathbf{x} \approx \gamma(\mathbf{M}_\mathbf{x}, \mathbf{S}_\mathbf{x}) \cdot \mathbf{A}_\ell$$

where $\approx$ refers to an additive error term that depends on $|\ \mathbf{D}_{i,b}\ |, |\ \mathbf{E}_{i,b}\ |, |\ \gamma(\mathbf{M}_{i,b}, \mathbf{S}_{i,b})\ |$, which we require to be small.

Semantic security. Following [CC17, GKW17b, GKW17a, WZ17], we consider the following notion of semantic security for γ-GGH15 encodings, namely that

> **(semantic security.)** The output $(\mathbf{A}_0, \{\mathbf{D}_{i,b}\}_{i\in[\ell],b\in\{0,1\}})$ computationally hides $\{\mathbf{M}_{i,b}\}_{i\in[\ell],b\in\{0,1\}}$. We only require that security holds "on average" over random $\{\mathbf{S}_{i,b}\}_{i\in[\ell],b\in\{0,1\}}, \mathbf{A}_\ell$.

Prior works [CC17, GKW17a, WZ17] showed that the $\gamma_\otimes$-GGH15 encodings achieve semantic security if we restrict the $\mathbf{M}_{i,b}$'s to be permutation matrices. That is,

> **Informal Lemma.** Under the LWE assumption, we have that for all *permutation* matrices $\{\mathbf{M}_{i,b}\}_{i\in[\ell],b\in\{0,1\}}$,
>
> $$(\mathbf{A}_0, \{\mathbf{D}_{i,0}, \mathbf{D}_{i,1}\}_{i\in[\ell]}) \approx_c (\mathbf{A}_0, \{\mathbf{V}_{i,0}, \mathbf{V}_{i,1}\}_{i\in[\ell]}) \tag{4}$$
>
> where $\mathbf{D}_{i,b} \leftarrow \mathbf{A}_{i-1}^{-1}((\mathbf{M}_{i,b} \otimes \mathbf{S}_{i,b})\mathbf{A}_i + \mathbf{E}_{i,b})$, and $\mathbf{V}_{i,0}, \mathbf{V}_{i,1}$ are random low-norm Gaussians.

As mentioned earlier in the introduction, the proof of security crucially relies on the fact that any permutation matrix $\mathbf{M}$, LWE tells us that $(\mathbf{A}, (\mathbf{M}\otimes\mathbf{S})\mathbf{A} + \mathbf{E}) \approx_c (\mathbf{A}, \mathbf{U})$, where $\mathbf{U}$ is uniformly random. We sketch the proof of the semantic security of $\gamma_\otimes$-GGH15 for $\ell = 1$, which extends readily to larger ℓ (here the major changes in the hybrid arguments are highlighted with boxes):

$$\begin{aligned}
& \left(\mathbf{A}_0, \{\mathbf{A}_0^{-1}((\mathbf{M}_{1,b}\otimes\mathbf{S}_{1,b})\mathbf{A}_1 + \mathbf{E}_{1,b})\}_{b\in\{0,1\}}\right) \\
\approx_c\ & \left(\mathbf{A}_0, \{\mathbf{A}_0^{-1}(\boxed{\mathbf{U}_{1,b}})\}_{b\in\{0,1\}}\right) \qquad \text{// LWE} \\
\approx_s\ & \left(\mathbf{A}_0, \{\boxed{\mathbf{V}_{1,b}}\}_{b\in\{0,1\}}\right) \qquad \text{// GPV}
\end{aligned}$$

2.2 This work: semantic security for arbitrary matrices

Without further modifications, γ-GGH15 encoding does not achieve semantic security for arbitrary matrices. Concretely, given $\mathbf{A}_0, \mathbf{D}_{1,0}$, we can compute

$$\mathbf{A}_0 \cdot \mathbf{D}_{1,0} = \gamma(\mathbf{M}_{1,0}, \mathbf{S}_{1,0})\mathbf{A}_1 + \mathbf{E}_{1,0}$$

which might leak information about the structure of $\mathbf{M}_{1,0}$. In particular, we can distinguish between $\mathbf{M}_{1,0}$ being $\mathbf{I}^{w\times w}$ versus $\mathbf{0}^{w\times w}$ for all of $\gamma_\times, \gamma_\otimes, \gamma_{\text{diag}}$.

The key to our new cryptographic constructions for general branching programs is a new technical lemma asserting semantic security for γ_{diag}-GGH15 encodings with arbitrary matrices where we replace $\mathbf{A}_0$ with $\mathbf{J}\mathbf{A}_0$ for some wide bookend matrix $\mathbf{J}$ that statistically "loses" information about $\mathbf{A}_0$:

New Lemma, Informal. Under the LWE assumption, we have that for all matrices $\{\mathbf{M}_{i,b}\}_{i\in[\ell],b\in\{0,1\}}$ over $\mathbb{Z}$,

$$(\mathbf{J}\mathbf{A}_0, (\mathbf{D}_{i,0}, \mathbf{D}_{i,1})_{i\in\ell}) \approx_c (\mathbf{J}\mathbf{A}_0, (\mathbf{V}_{i,0}, \mathbf{V}_{i,1})_{i\in\ell}) \tag{5}$$

where $\mathbf{J}$ is any matrix of the form $[\star \mid \mathbf{I}]$, $\mathbf{D}_{i,b} \leftarrow \mathbf{A}_{i-1}^{-1}(\begin{pmatrix}\mathbf{M}_{i,b} & \\ & \mathbf{S}_{i,b}\end{pmatrix}\mathbf{A}_i + \mathbf{E}_{i,b})$, and $\mathbf{V}_{i,0}, \mathbf{V}_{i,1}$ are random low-norm Gaussians.

2.2.1 New proof technique

We prove a stronger statement for the semantic security of γ_{diag}-GGH15, namely the semantic security holds even given $\mathbf{S}_{1,0}, \mathbf{S}_{1,1}, \ldots, \mathbf{S}_{\ell,0}, \mathbf{S}_{\ell,1}$ (but not $\mathbf{A}_1, \ldots, \mathbf{A}_\ell$). Our proof departs significantly from the prior analysis – in particular, we will treat $\mathbf{A}_1, \ldots, \mathbf{A}_\ell$ as LWE secrets. Let $\overline{\mathbf{A}}_i, \underline{\mathbf{A}}_i$ denote the top and bottom parts of $\mathbf{A}$, and define $\overline{\mathbf{E}}_{i,b}, \underline{\mathbf{E}}_{i,b}$ analogously. This means that

$$\mathbf{A}_{i-1}^{-1}(\gamma_{\text{diag}}(\mathbf{M}_{i,b}, \mathbf{S}_{i,b})\mathbf{A}_i + \mathbf{E}_{i,b}) = \mathbf{A}_{i-1}^{-1}\begin{pmatrix}\mathbf{M}_{i,b}\overline{\mathbf{A}}_i + \overline{\mathbf{E}}_{i,b} \\ \mathbf{S}_{i,b}\underline{\mathbf{A}}_i + \underline{\mathbf{E}}_{i,b}\end{pmatrix}$$

We will use $\mathbf{A}_1, \ldots, \mathbf{A}_\ell$ as LWE secrets in the following order: $\underline{\mathbf{A}}_\ell, \ldots, \underline{\mathbf{A}}_1, \overline{\mathbf{A}}_0, \ldots, \overline{\mathbf{A}}_{\ell-1}$. We sketch the proof for $\ell = 1$ (and it extends readily to larger ℓ):

$$\begin{aligned}
& \left(\mathbf{J}\mathbf{A}_0, \{\mathbf{A}_0^{-1}\begin{pmatrix}\mathbf{M}_{1,b}\overline{\mathbf{A}}_1 + \overline{\mathbf{E}}_{1,b} \\ \mathbf{S}_{1,b}\underline{\mathbf{A}}_1 + \underline{\mathbf{E}}_{1,b}\end{pmatrix}\}_{b\in\{0,1\}}\right) \\
\approx_c \quad & \left(\mathbf{J}\mathbf{A}_0, \{\boxed{\overline{\mathbf{A}}_0^{-1}\left(\mathbf{M}_{1,b}\overline{\mathbf{A}}_1 + \overline{\mathbf{E}}_{1,b}\right)}\}_{b\in\{0,1\}}\right) \\
\approx_s \quad & \left(\boxed{\mathbf{U}_0}, \{\overline{\mathbf{A}}_0^{-1}\left(\mathbf{M}_{1,b}\overline{\mathbf{A}}_1 + \overline{\mathbf{E}}_{1,b}\right)\}_{b\in\{0,1\}}\right) \\
\approx_c \quad & \left(\mathbf{U}_0, \{\boxed{\mathbf{V}_{1,b}}\}_{b\in\{0,1\}}\right)
\end{aligned}$$

where the notations and analysis of hybrid arguments are as follows

- The first $\approx_c$ follow from a more general statement, namely for all i and for any $\mathbf{Z}_{i,b}$, we have

$$\left\{\mathbf{A}_{i-1}^{-1}\begin{pmatrix}\mathbf{Z}_{i,b} \\ \mathbf{S}_{i,b}\underline{\mathbf{A}}_i + \underline{\mathbf{E}}_{i,b}\end{pmatrix}\right\}_{b\in\{0,1\}} \approx_c \left\{\overline{\mathbf{A}}_{i-1}^{-1}\left(\mathbf{Z}_{i,b}\right)\right\}_{b\in\{0,1\}}$$

  even if the distinguisher gets $\mathbf{A}_{i-1}, \mathbf{S}_{i,b}, \mathbf{Z}_{i,b}$. The proof of this statement follows by first applying LWE with $\underline{\mathbf{A}}_i$ as the secret[4] to deduce that

$$\{\mathbf{S}_{i,b}, \mathbf{S}_{i,b}\underline{\mathbf{A}}_i + \underline{\mathbf{E}}_{i,b}\}_{b\in\{0,1\}} \approx_c \{\mathbf{S}_{i,b}, \mathbf{U}_{i,b}\}_{b\in\{0,1\}}$$

  where the $\mathbf{U}_{i,b}$ matrices are uniformly random over $\mathbb{Z}_q$, followed by a new statistical lemma about trapdoor sampling which tells us that for all but negligibly many $\mathbf{A}_{i-1}$, we have that for all $\mathbf{Z}_{i,b}$,

$$\mathbf{A}_{i-1}^{-1}\begin{pmatrix}\mathbf{Z}_{i,b} \\ \mathbf{U}_{i,b}\end{pmatrix} \approx_s \overline{\mathbf{A}}_{i-1}^{-1}\left(\mathbf{Z}_{i,b}\right)$$

[4] Here, we could have used $\mathbf{S}_{i,0}, \mathbf{S}_{i,1}$ as the LWE secrets and $\underline{\mathbf{A}}_i$ as the public matrix; however, this strategy would break down when $\mathbf{M}_{i,b}$ depends on $\mathbf{S}_{i,b}$, which is needed in the applications.

- The $\approx_s$ follows from the structure of $\mathbf{J}$, which implies $(\overline{\mathbf{A}}_0, \mathbf{J}\mathbf{A}_0) \approx_s (\overline{\mathbf{A}}_0, \mathbf{U}_0)$, where $\mathbf{U}_0$ is a uniformly random matrix.
- The final $\approx_c$ follows from a more general statement, which says that under the LWE assumption, we have that for any $\mathbf{Z}$,
$$\mathbf{A}^{-1}(\mathbf{Z} + \mathbf{E}) \approx_c \mathbf{A}^{-1}(\mathbf{U})$$
where the distributions are over random choices of $\mathbf{A}, \mathbf{E}, \mathbf{U}$, provided $\mathbf{A}$ is hidden from the distinguisher. The proof uses the Bonsai technique [CHKP12]. Suppose $\mathbf{A}$ is of the form $[\mathbf{A}_1 \mid \mathbf{A}_2]$ where $\mathbf{A}_1$ is uniformly random, $\mathbf{A}_2$ sampled with a trapdoor. Then, we have via the Bonsai technique [CHKP12]:
$$\mathbf{A}^{-1}(\mathbf{Z} + \mathbf{E}) \approx_s \begin{pmatrix} -\mathbf{V} \\ \mathbf{A}_2^{-1}(\mathbf{A}_1\mathbf{V} + \mathbf{E} + \mathbf{Z})) \end{pmatrix}$$
where $\mathbf{V}$ is a random low-norm Gaussian. We then apply the LWE assumption to $(\mathbf{V}, \mathbf{A}_1\mathbf{V} + \mathbf{E})$ with $\mathbf{A}_1$ as the LWE secret. Once we replace $\mathbf{A}_1\mathbf{V} + \mathbf{E}$ with a uniformly random matrix, the rest of the proof follows readily from the standard GPV lemma.

Extension: combining $\gamma_{\otimes}, \gamma_{\mathbf{diag}}$. For the applications to private constrained PRFs and lockable obfuscation, we will rely on $\gamma_{\otimes\mathsf{diag}}$-GGH15 encodings, where
$$\gamma_{\otimes\mathsf{diag}}(\mathbf{M}, \mathbf{S}) := \begin{pmatrix} \mathbf{M} \otimes \mathbf{S} & \\ & \mathbf{S} \end{pmatrix}$$
We observe that our proof of semantic security for γ_{diag} also implies semantic security for $\gamma_{\otimes\mathsf{diag}}$, where we give out $\mathbf{J}\mathbf{A}_0$ instead of $\mathbf{A}_0$. This follows from the fact that our proof for γ_{diag} goes through even if the $\mathbf{M}_{i,b}$'s depend on the $\mathbf{S}_{i,b}$'s, since we treat the latter as public matrices when we invoke the LWE assumption.

2.3 New Cryptographic Constructions from LWE

Using $\gamma_{\otimes\mathsf{diag}}$-GGH15 encodings and the proof that semantic security of $\gamma_{\otimes\mathsf{diag}}$ holds for arbitrary $\mathbf{M}$ matrices, we are ready to construct private constrained PRFs and lockable obfuscation where the constraint/function can be recognized by arbitrary matrix branching programs. Here we briefly explain the private constrained PRF construction as an example.

Before that we recall some terminologies for matrix branching programs. In the overview, we focus on read-once matrix branching programs for notational simplicity, although our scheme works for general matrix branching programs with any input pattern and matrix pattern (possibly low-rank matrices). A (read-once) matrix branching program for a function $f_\Gamma : \{0,1\}^\ell \to \{0,1\}$ is specified by $\Gamma := \left\{ \{\mathbf{M}_{i,b}\}_{i\in[\ell], b\in\{0,1\}}, \mathbf{P}_0, \mathbf{P}_1 \right\}$ such that for all $\mathbf{x} \in \{0,1\}^\ell$,
$$\mathbf{M}_{\mathbf{x}} = \prod_{i=1}^{\ell} \mathbf{M}_{i,x_i} = \mathbf{P}_{f_\Gamma(x)}$$
We will work with families of branching programs $\{\Gamma\}$, which share the same $\mathbf{P}_0, \mathbf{P}_1$.

2.3.1 Private constrained PRFs

We proceed to provide an overview of our construction of private constrained PRFs using $\gamma_{\otimes\text{diag}}$-GGH15 encodings. As a quick overview of a private constrained PRF, a private constrained PRF allows the PRF master secret key holder to derive a constrained key given a constraint predicate C. The constrained key is required to randomize the output on every input x s.t. $C(x) = 0$, preserve the output on every input x s.t. $C(x) = 1$. In addition, the constraint C is required to be hidden given the description of the constrained key.

Let $\mathbf{e}_i \in \{0,1\}^{1\times w}$ denotes the unit vector with the i^{th} coordinate being 1, the rest being 0. Consider a class of constraints recognizable by branching programs

$$\Gamma_C := \left\{ \left\{\mathbf{M}_{i,b} \in \{0,1\}^{w\times w}\right\}_{i\in[\ell],b\in\{0,1\}}, \mathbf{P}_0, \mathbf{P}_1 \right\},$$

where the target matrices $\mathbf{P}_0, \mathbf{P}_1$ satisfy $\mathbf{e}_1\mathbf{P}_0 = \mathbf{e}_1$, $\mathbf{e}_1\mathbf{P}_1 = \mathbf{0}^{1\times w}$.

We use $\gamma_{\otimes\text{diag}}$ to encode $\{\mathbf{M}_{i,b}\}_{i\in[\ell],b\in\{0,1\}}$, which means for $i = 0, ..., \ell$, $\mathbf{A}_i \in \mathbb{Z}_q^{(nw+n)\times m}$. Denote $\underline{\mathbf{A}}_0$ as the bottom n rows of $\mathbf{A}_i$, $\overline{\mathbf{A}}_i$ as the top nw rows of $\mathbf{A}_i$. Inside $\overline{\mathbf{A}}_i$ let $\overline{\mathbf{A}}_i^{(j)}$ denote the $(j-1)n^{th}$ to jn^{th} rows of $\overline{\mathbf{A}}_i$, for $j \in [w]$.

Define the output of the normal PRF evaluation as

$$\mathbf{x} \mapsto \lfloor \mathbf{S}_\mathbf{x}\underline{\mathbf{A}}_\ell \rceil_p$$

where $\lfloor \cdot \rceil_p$ denotes the rounding-to-$\mathbb{Z}_p$ operation used in previous LWE-based PRFs, which we suppress in the rest of this overview for notational simplicity.

We set $\mathbf{J} := (\mathbf{e}_1 \otimes \mathbf{I} \mid \mathbf{I})$ so that $\mathbf{J}\cdot\mathbf{A}_0 = \overline{\mathbf{A}}_0^{(1)} + \underline{\mathbf{A}}_0$, then

$$\mathbf{J}\cdot\gamma_{\otimes\text{diag}}(\mathbf{M}_\mathbf{x}, \mathbf{S}_\mathbf{x})\cdot\mathbf{A}_\ell = ((\mathbf{e}_1\cdot\mathbf{M}_\mathbf{x})\otimes\mathbf{S}_\mathbf{x})\cdot\overline{\mathbf{A}}_\ell + \mathbf{S}_\mathbf{x}\underline{\mathbf{A}}_\ell = \begin{cases} \mathbf{S}_\mathbf{x}\underline{\mathbf{A}}_\ell & \text{if } f_\Gamma(\mathbf{x}) = 1 \\ \mathbf{S}_\mathbf{x}\overline{\mathbf{A}}_\ell^{(1)} + \mathbf{S}_\mathbf{x}\underline{\mathbf{A}}_\ell & \text{if } f_\Gamma(\mathbf{x}) = 0 \end{cases}$$

Given Γ, the constrained key is constructed as

$$\overline{\mathbf{A}}_0^{(1)} + \underline{\mathbf{A}}_0, (\mathbf{D}_{i,0}, \mathbf{D}_{i,0})_{i\in[\ell]}$$

where $(\mathbf{A}_0, \{\mathbf{D}_{i,b}\}_{i\in[\ell],b\in\{0,1\}}) \leftarrow \text{GGHEnc}_{\otimes\text{diag}}(\{\mathbf{M}_{i,b}\}_{i\in[\ell],b\in\{0,1\}}, \{\mathbf{S}_{i,b}\}_{i\in[\ell],b\in\{0,1\}}, \mathbf{A}_\ell)$.

The constrained evaluation on an input $\mathbf{x}$ gives

$$(\overline{\mathbf{A}}_0^{(1)} + \underline{\mathbf{A}}_0)\cdot\mathbf{D}_\mathbf{x} \approx \mathbf{J}\cdot\gamma_{\otimes\text{diag}}(\mathbf{M}_\mathbf{x}, \mathbf{S}_\mathbf{x})\cdot\mathbf{A}_\ell$$

which equals to $\mathbf{S}_\mathbf{x}\underline{\mathbf{A}}_\ell$ if $f_\Gamma(\mathbf{x}) = 1$, $\mathbf{S}_\mathbf{x}\overline{\mathbf{A}}_\ell^{(1)} + \mathbf{S}_\mathbf{x}\underline{\mathbf{A}}_\ell$ if $f_\Gamma(\mathbf{x}) = 0$.

A special case: private puncturable PRFs. A private puncturable PRF can be obtained by simply using branching program with 1×1 matrices (i.e. let $w = 1$). The punctured key at $\mathbf{x}^*$ is given by

$$\overline{\mathbf{A}}_0 + \underline{\mathbf{A}}_0, \{\mathbf{D}_{i,b}\}_{i\in[\ell],b\in\{0,1\}}$$

where

$$\mathbf{D}_{i,x_i^*} \leftarrow \mathbf{A}_{i-1}^{-1}\left(\begin{pmatrix}\mathbf{S}_{i,x_i^*} & \\ & \mathbf{S}_{i,x_i^*}\end{pmatrix}\mathbf{A}_i + \mathbf{E}_{i,x_i^*}\right), \mathbf{D}_{i,1-x_i^*} \leftarrow \mathbf{A}_{i-1}^{-1}\left(\begin{pmatrix}\mathbf{0} & \\ & \mathbf{S}_{i,1-x^*}\end{pmatrix}\mathbf{A}_i + \mathbf{E}_{i,1-x_i^*}\right).$$

The construction extends naturally to allow us to puncture at sets of points specified by a wildcard pattern $\{0,1,\star\}^\ell$.

Security. In the security proof, we will use the fact that whenever $f_\Gamma(\mathbf{x}) = 0$, constrained evaluation outputs $\boxed{\mathbf{S}_\mathbf{x}\overline{\mathbf{A}}_\ell^{(1)}} + \mathbf{S}_\mathbf{x}\underline{\mathbf{A}}_\ell$, so that the normal PRF output is masked by the boxed term. More formally, in the security game, the adversary gets a constrained key for Γ_C, and oracle access to a PRF evaluation oracle Eval. We consider the following sequence of games:

- Replace the output of the Eval oracle by

$$(\overline{\mathbf{A}}_0^{(1)} + \underline{\mathbf{A}}_0) \cdot \mathbf{D}_\mathbf{x} - \mathbf{S}_\mathbf{x} \cdot \overline{\mathbf{A}}_\ell^{(1)}$$

  This is statistically indistinguishable from the real game, since $(\overline{\mathbf{A}}_0^{(1)} + \underline{\mathbf{A}}_0) \cdot \mathbf{D}_\mathbf{x} \approx \mathbf{S}_\mathbf{x} \cdot \overline{\mathbf{A}}_\ell^{(1)} + \mathbf{S}_\mathbf{x} \cdot \underline{\mathbf{A}}_\ell$, and the approximation disappears w.h.p. after rounding.

- Apply semantic security to replace $(\mathbf{D}_{i,0}, \mathbf{D}_{i,0})_{i\in[\ell]}$ with random. Here, we require that semantic security holds even if the distinguisher gets $\{\mathbf{S}_{i,b}\}_{i\in[\ell],b\in\{0,1\}}, \overline{\mathbf{A}}_\ell$, where the latter are needed in order to compute $\mathbf{S}_\mathbf{x} \cdot \overline{\mathbf{A}}_\ell^{(1)}$. This implies constraint-hiding.

- Now, we can apply the BLMR analysis to deduce pseudorandomness of $\mathbf{S}_\mathbf{x} \cdot \overline{\mathbf{A}}_\ell^{(1)}$, where we treat $\overline{\mathbf{A}}_\ell^{(1)}$ as the seed of the BLMR PRF [BLMR13]. This implies pseudorandomness of the output of the Eval oracle.

3 Preliminaries

Notations and terminology. In cryptography, the security parameter (denoted as λ) is a variable that is used to parameterize the computational complexity of the cryptographic algorithm or protocol, and the adversary's probability of breaking security. An algorithm is "efficient" if it runs in (probabilistic) polynomial time over λ.

When a variable v is drawn randomly from the set S we denote as $v \xleftarrow{\$} S$ or $v \leftarrow U(S)$, sometimes abbreviated as v when the context is clear. We use $\approx_s$ and $\approx_c$ as the abbreviation for statistically close and computationally indistinguishable.

Let $\mathbb{R}, \mathbb{Z}, \mathbb{N}$ be the set of real numbers, integers and positive integers. Denote $\mathbb{Z}/(q\mathbb{Z})$ by $\mathbb{Z}_q$. The rounding operation $\lfloor a \rceil_p : \mathbb{Z}_q \to \mathbb{Z}_p$ is defined as multiplying a by p/q and rounding the result to the nearest integer.

For $n \in \mathbb{N}$, $[n] := \{1, ..., n\}$. A vector in $\mathbb{R}^n$ (represented in column form by default) is written as a bold lower-case letter, e.g. $\mathbf{v}$. For a vector $\mathbf{v}$, the i^{th} component of $\mathbf{v}$ will be denoted by v_i. A matrix is written as a bold capital letter, e.g. $\mathbf{A}$. The i^{th} column vector of $\mathbf{A}$ is denoted $\mathbf{a}_i$. In this article we frequently meet the situation where a matrix $\mathbf{A}$ is partitioned into two pieces, one stacking over the other. We denote it as $\mathbf{A} = \binom{\overline{\mathbf{A}}}{\underline{\mathbf{A}}}$. The partition is not necessarily even. We will explicitly mention the dimension when needed.

The length of a vector is the ℓ_p-norm $\|\mathbf{v}\|_p = (\sum v_i^p)^{1/p}$. The length of a matrix is the norm of its longest column: $\|\mathbf{A}\|_p = \max_i \|\mathbf{a}_i\|_p$. By default we use ℓ_2-norm unless explicitly mentioned. When a vector or matrix is called "small", we refer to its norm.

Subset products (of matrices) appear frequently in this article. For a given $h \in \mathbb{N}$, a bit-string $\mathbf{v} \in \{0,1\}^h$, we use $\mathbf{X}_\mathbf{v}$ to denote $\prod_{i\in[h]} \mathbf{X}_{i,v_i}$ (it is implicit that $\{\mathbf{X}_{i,b}\}_{i\in[h],b\in\{0,1\}}$ are well-defined).

The tensor product (Kronecker product) for matrices $\mathbf{A} \in \mathbb{R}^{\ell \times m}$, $\mathbf{B} \in \mathbb{R}^{n \times p}$ is defined as

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{1,1}\mathbf{B}, & \ldots, & a_{1,m}\mathbf{B} \\ \ldots, & \ldots, & \ldots \\ a_{\ell,1}\mathbf{B}, & \ldots, & a_{\ell,m}\mathbf{B} \end{bmatrix} \in \mathbb{R}^{\ell n \times mp}. \tag{6}$$

The rank of the resultant matrix satisfies $\mathsf{rank}(\mathbf{A} \otimes \mathbf{B}) = \mathsf{rank}(\mathbf{A}) \cdot \mathsf{rank}(\mathbf{B})$.
For matrices $\mathbf{A} \in \mathbb{R}^{\ell \times m}$, $\mathbf{B} \in \mathbb{R}^{n \times p}$, $\mathbf{C} \in \mathbb{R}^{m \times u}$, $\mathbf{D} \in \mathbb{R}^{p \times v}$,

$$(\mathbf{AC}) \otimes (\mathbf{BD}) = (\mathbf{A} \otimes \mathbf{B}) \cdot (\mathbf{C} \otimes \mathbf{D}). \tag{7}$$

Lemma 3.1 (Leftover hash lemma). *Let $\mathcal{H} = \{h : \mathcal{X} \to \mathcal{Y}\}$ be a 2-universal hash function family. Then for any random variable $X \in \mathcal{X}$, for $\epsilon > 0$ s.t. $\log(|\mathcal{Y}|) \leq H_\infty(X) - 2\log(1/\epsilon)$, the distributions*

$$(h, h(X)) \text{ and } (h, U(\mathcal{Y}))$$

are ϵ-statistically close.

3.1 Lattices background

An n-dimensional lattice Λ is a discrete additive subgroup of $\mathbb{R}^n$. Given n linearly independent basis vectors $\mathbf{B} = \{\mathbf{b}_1, ..., \mathbf{b}_n \in \mathbb{R}^n\}$, the lattice generated by $\mathbf{B}$ is $\Lambda(\mathbf{B}) = \Lambda(\mathbf{b}_1, ..., \mathbf{b}_n) = \{\sum_{i=1}^{n} x_i \cdot \mathbf{b}_i, x_i \in \mathbb{Z}\}$. Let $\tilde{\mathbf{B}}$ denote the Gram-Schmidt orthogonalization of $\mathbf{B}$.

Gaussian on lattices. For any $\sigma > 0$ define the Gaussian function on $\mathbb{R}^n$ centered at $\mathbf{c}$ with parameter σ:

$$\forall \mathbf{x} \in \mathbb{R}^n,\ \rho_{\sigma,\mathbf{c}}(\mathbf{x}) = e^{-\pi \|\mathbf{x}-\mathbf{c}\|^2/\sigma^2}$$

The subscripts σ and $\mathbf{c}$ are taken to be 1 and $\mathbf{0}$ (respectively) when omitted.

For any $\mathbf{c} \in \mathbb{R}^n$, real $\sigma > 0$, and n-dimensional lattice Λ, define the discrete Gaussian distribution over Λ as:

$$\forall \mathbf{x} \in \Lambda,\ D_{\Lambda,\sigma,\mathbf{c}}(\mathbf{x}) = \frac{\rho_{\sigma,\mathbf{c}}(\mathbf{x})}{\rho_{\sigma,\mathbf{c}}(\Lambda)}$$

Lemma 3.2 (Noise smudging [DGK+10]). *Let $y, \sigma \in \mathbb{R}^+$. The statistical distance between the distributions $D_{\mathbb{Z},\sigma}$ and $D_{\mathbb{Z},\sigma+y}$ is at most y/σ.*

Smoothing parameter. We recall the definition of smoothing parameter and some useful facts.

Definition 1 (Smoothing parameter [MR07]). For any n-dimensional lattice Λ and positive real $\epsilon > 0$, the smoothing parameter $\eta_\epsilon(\Lambda)$ is the smallest real $\sigma > 0$ such that $\rho_{1/\sigma}(\Lambda^* \setminus \{\mathbf{0}\}) \leq \epsilon$.

Lemma 3.3 (Smoothing parameter bound from [GPV08]). *For any n-dimensional lattice $\Lambda(\mathbf{B})$ and for any $\omega(\sqrt{\log n})$ function, there is a negligible $\epsilon(n)$ for which*

$$\eta_\epsilon(\Lambda) \leq \|\tilde{\mathbf{B}}\| \cdot \omega(\sqrt{\log n})$$

Lemma 3.4 (Smooth over the cosets [GPV08]). *Let Λ, Λ' be n-dimensional lattices s.t. $\Lambda' \subseteq \Lambda$. Then for any $\epsilon > 0$, $\sigma > \eta_\epsilon(\Lambda')$, and $\mathbf{c} \in \mathbb{R}^n$, we have*

$$\Delta(D_{\Lambda,\sigma,\mathbf{c}} \bmod \Lambda',\ U(\Lambda \bmod \Lambda')) < 2\epsilon$$

Lemma 3.5 ([PR06, MR07]). *Let $\mathbf{B}$ be a basis of an n-dimensional lattice Λ, and let $\sigma \geq \|\tilde{\mathbf{B}}\| \cdot \omega(\log n)$, then $\Pr_{\mathbf{x} \leftarrow D_{\Lambda,\sigma}}[\|\mathbf{x}\| \geq \sigma \cdot \sqrt{n} \vee \mathbf{x} = \mathbf{0}] \leq \mathrm{negl}(n)$.*

Gentry, Peikert and Vaikuntanathan [GPV08] show how to sample statistically close to discrete Gaussian distribution in polynomial time for sufficiently large σ (the algorithm is first proposed by Klein [Kle00]). The sampler is upgraded in [BLP+13] so that the output is distributed exactly as a discrete Gaussian.

Lemma 3.6 ([GPV08, BLP+13]). *There is a p.p.t. algorithm that, given a basis $\mathbf{B}$ of an n-dimensional lattice $\Lambda(\mathbf{B})$, $\mathbf{c} \in \mathbb{R}^n$, $\sigma \geq \|\tilde{\mathbf{B}}\| \cdot \sqrt{\ln(2n+4)/\pi}$, outputs a sample from $D_{\Lambda,\sigma,\mathbf{c}}$.*

Learning with errors. We recall the learning with errors problem.

Definition 2 (Decisional learning with errors (LWE) [Reg09]). For $n, m \in \mathbb{N}$ and modulus $q \geq 2$, distributions for secret vectors, public matrices, and error vectors $\theta, \pi, \chi \subseteq \mathbb{Z}_q$. An LWE sample is obtained from sampling $\mathbf{s} \leftarrow \theta^n$, $\mathbf{A} \leftarrow \pi^{n \times m}$, $\mathbf{e} \leftarrow \chi^m$, and outputting $(\mathbf{A}, \mathbf{s}^T\mathbf{A} + \mathbf{e}^T \pmod q)$.

We say that an algorithm solves $\mathsf{LWE}_{n,m,q,\theta,\pi,\chi}$ if it distinguishes the LWE sample from a random sample distributed as $\pi^{n \times m} \times U(\mathbb{Z}_q^{1 \times m})$ with probability bigger than $1/2$ plus non-negligible.

Lemma 3.7 (Regularity of Ajtai function [Reg09]). *Fix a constant $c > 1$, let $m \geq cn \log q$. Then for all but $q^{\frac{-(c-1)n}{4}}$ fraction of $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, the statistical distance between a random subset-sum of the columns of $\mathbf{A}$ and uniform over $\mathbb{Z}_q^n$ is less than $q^{\frac{-(c-1)n}{4}}$.*

Lemma 3.8 (Standard form [Reg09, Pei09, BLP+13, PRS17]). *Given $n \in \mathbb{N}$, for any $m = \mathrm{poly}(n)$, $q \leq 2^{\mathrm{poly}(n)}$. Let $\theta = \pi = U(\mathbb{Z}_q)$, $\chi = D_{\mathbb{Z},\sigma}$ where $\sigma \geq 2\sqrt{n}$. If there exists an efficient (possibly quantum) algorithm that breaks $\mathsf{LWE}_{n,m,q,\theta,\pi,\chi}$, then there exists an efficient (possibly quantum) algorithm for approximating SIVP and GapSVP in the ℓ_2 norm, in the worst case, to within $\tilde{O}(nq/\sigma)$ factors.*

We drop the subscripts of LWE when referring to standard form of LWE with the parameters specified in Lemma 3.8. In this article we frequently use the following variant of LWE that is implied by the standard form.

Lemma 3.9 (LWE with small public matrices [BLMR13]). *For n, m, q, σ chosen as was in Lemma 3.8, $\mathsf{LWE}_{n',m,q,U(\mathbb{Z}_q),D_{\mathbb{Z},\sigma},D_{\mathbb{Z},\sigma}}$ is as hard as $\mathsf{LWE}_{n,m,q,U(\mathbb{Z}_q),U(\mathbb{Z}_q),D_{\mathbb{Z},\sigma}}$ for $n' \geq 2 \cdot n \log q$.*

Trapdoor and preimage sampling. Given $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, denote the kernel lattice of $\mathbf{A}$ as

$$\Lambda^\perp(\mathbf{A}) := \{\mathbf{c} \in \mathbb{Z}^m : \mathbf{A} \cdot \mathbf{c} = 0^n \pmod q\}.$$

Given any $\mathbf{y} \in \mathbb{Z}_q^n$, $\sigma > 0$, we use $\mathbf{A}^{-1}(\mathbf{y}, \sigma)$ to denote the distribution of a vector $\mathbf{d}$ sampled from $D_{\mathbb{Z}^m,\sigma}$ conditioned on $\mathbf{A}\mathbf{d} = \mathbf{y} \pmod q$. We sometimes suppress σ when the context is clear.

Lemma 3.10 ([Ajt99, AP11, MP12]). *There is a p.p.t. algorithm $\mathsf{TrapSam}(1^n, 1^m, q)$ that, given the modulus $q \geq 2$, dimensions n, m such that $m \geq 2n \log q$, outputs $\mathbf{A} \approx_s U(\mathbb{Z}_q^{n \times m})$ with a trapdoor τ.*

Following Lemmas 3.6 and 3.10,

Lemma 3.11. *There is a p.p.t. algorithm that for $\sigma \geq 2\sqrt{n \log q}$, given $(\mathbf{A}, \tau) \leftarrow \mathsf{TrapSam}(1^n, 1^m, q)$, $\mathbf{y} \in \mathbb{Z}_q^n$, outputs a sample from $\mathbf{A}^{-1}(\mathbf{y}, \sigma)$.*

Lemma 3.12 ([GPV08])**.** *For all but negligible probability over $(\mathbf{A}, \tau) \leftarrow \mathsf{TrapSam}(1^n, 1^m, q)$, for sufficiently large $\sigma \geq 2\sqrt{n \log q}$, the following distributions are efficiently samplable and statistically close:*

$$\left\{\mathbf{A}, \mathbf{x}, \mathbf{y} : \mathbf{y} \leftarrow U(\mathbb{Z}_q^n), \mathbf{x} \leftarrow \mathbf{A}^{-1}(\mathbf{y}, \sigma)\right\} \approx_s \left\{\mathbf{A}, \mathbf{x}, \mathbf{y} : \mathbf{x} \leftarrow D_{\mathbb{Z}^m, \sigma}, \mathbf{y} = \mathbf{A}\mathbf{x}\right\}.$$

Lemma 3.13 (Bonsai technique [CHKP12])**.** *Let $n, m, m_1, m_2, q \in \mathbb{N}, \sigma \in \mathbb{R}$ satisfy $m = m_1 + m_2$, $m_2 \geq 2n \log q$, $\sigma > 2\sqrt{n \log q}$. For any $\mathbf{y} \in \mathbb{Z}_q^n$, the following two distributions are efficiently samplable and statistically close.*

1. *Let $(\mathbf{A}, \tau) \leftarrow \mathsf{TrapSam}(1^n, 1^m, q)$, $\mathbf{d} \leftarrow \mathbf{A}^{-1}(\mathbf{y}, \sigma)$. Output $(\mathbf{A}, \mathbf{d})$.*
2. *Let $\mathbf{A}_1 \leftarrow U(\mathbb{Z}_q^{n \times m_1})$, $(\mathbf{A}_2, \tau_2) \leftarrow \mathsf{TrapSam}(1^n, 1^{m_2}, q)$; $\mathbf{d}_1 \leftarrow D_{\mathbb{Z}^{m_1}, \sigma}$, $\mathbf{d}_2 \leftarrow \mathbf{A}_2^{-1}(\mathbf{y} - \mathbf{A}_1 \cdot \mathbf{d}_1, \sigma)$. Let $\mathbf{A} = [\mathbf{A}_1, \mathbf{A}_2]$, $\mathbf{d} = [\mathbf{d}_1^T, \mathbf{d}_2^T]^T$. Output $(\mathbf{A}, \mathbf{d})$.*

4 New Lemmas on Preimage Sampling

In this section, we present new lemmas related to lattice preimage sampling. These lemmas are essential to the proof of semantic security for non-permutation branching programs, as outlined in Section 2.2.

The first is a statistical lemma which states that for all but negligibly many matrix $\mathbf{A}$ (with proper dimensions), for any matrix $\mathbf{Z}$, the following two distributions are statistically indistinguishable:

$$\left(\mathbf{A}, \mathbf{A}^{-1}\begin{pmatrix}\mathbf{Z} \\ \mathbf{U}\end{pmatrix}\right) \approx_s \left(\mathbf{A}, \overline{\mathbf{A}}^{-1}(\mathbf{Z})\right)$$

where the distributions are over random choices of a matrix $\mathbf{U}$ and probability distributions $\mathbf{A}^{-1}(\cdot)$ and $\overline{\mathbf{A}}^{-1}(\cdot)$. This is in essence an extension of the trapdoor sampling lemma from Gentry, Peikert and Vaikuntanathan [GPV08].

The second is a computational lemma which states that for any matrix $\mathbf{Z}$, the following two distributions are computationally indistinguishable:

$$\mathbf{A}^{-1}(\mathbf{Z} + \mathbf{E}) \approx_c \mathbf{A}^{-1}(\mathbf{U})$$

where the distributions are over random private choices of $\mathbf{A}, \mathbf{E}$ and $\mathbf{U}$ and the coins of $\mathbf{A}^{-1}(\cdot)$. The computational indistinguishability relies on the hardness of the decisional learning with errors (LWE) problem.

4.1 The Statistical Lemma

We prove the above statistical lemma for vectors; the setting for matrices follow readily via a hybrid argument.

Lemma 4.1. *Let $\epsilon > 0$. Given $\sigma \in R^+$, $n', n, m, q \in \mathbb{N}$. For all but a $q^{-2n'}$ fraction of $\overline{\mathbf{A}} \in \mathbb{Z}_q^{n' \times m}$, all but a q^{-2n} fraction of $\underline{\mathbf{A}} \in \mathbb{Z}_q^{n \times m}$, let $\mathbf{A} := \binom{\overline{\mathbf{A}}}{\underline{\mathbf{A}}}$. For $\sigma > \eta_\epsilon(\Lambda^\perp(\mathbf{A}))$, $m \geq 9(n'+n)\log q$. For a fixed $\mathbf{z} \in \mathbb{Z}_q^{n'}$, for $\mathbf{u} \leftarrow U(\mathbb{Z}_q^n)$, we have*

$$\mathbf{A}^{-1}\left(\begin{pmatrix}\mathbf{z}\\ \mathbf{u}\end{pmatrix}, \sigma\right) \text{ and } \overline{\mathbf{A}}^{-1}(\mathbf{z}, \sigma)$$

are 2ϵ-statistically close.

Proof. We need two lemmas to assist the proof of Lemma 4.1.

Lemma 4.2. *Let $c > 9$. For $n', n, m, q \in \mathbb{N}$ such that $m \geq c(n'+n)\log q$. For all but $q^{-2n'}$ fraction of $\overline{\mathbf{A}} \in \mathbb{Z}_q^{n' \times m}$, all but q^{-2n} fraction of $\underline{\mathbf{A}} \in \mathbb{Z}_q^{n \times m}$, we have $\left\{\underline{\mathbf{A}} \cdot \mathbf{x} \mid \mathbf{x} \in \{0,1\}^m \cap \Lambda^\perp(\overline{\mathbf{A}})\right\} = \mathbb{Z}_q^n$.*

Proof. From Lemma 3.7, we have for all but $q^{-2n'}$ fraction of $\overline{\mathbf{A}} \in \mathbb{Z}_q^{n' \times m}$

$$\left|\Pr_{\mathbf{x} \in \{0,1\}^m}[\overline{\mathbf{A}} \cdot \mathbf{x} = 0^{n'}] - q^{-n'}\right| < 2q^{-2n'} \Rightarrow \Pr_{\mathbf{x} \in \{0,1\}^m}[\overline{\mathbf{A}} \cdot \mathbf{x} = 0^{n'}] > 0.99 \cdot q^{-n'} \tag{8}$$

Let $\mathbf{x} \leftarrow U(\{0,1\}^m \cap \Lambda^\perp(\overline{\mathbf{A}}))$, we have $H_\infty(\mathbf{x}) > m - 2n'\log q$. For $\delta > 0$, by setting $m \geq n\log q + 2n'\log q + 2\log(1/\delta)$, we have that for $\underline{\mathbf{A}} \leftarrow U(\mathbb{Z}_q^{n \times m})$,

$$(\underline{\mathbf{A}}, \underline{\mathbf{A}} \cdot \mathbf{x}) \text{ and } (\underline{\mathbf{A}}, U(\mathbb{Z}_q^n))$$

are δ-statistically close following leftover hash lemma (cf. Lemma 3.1).

Then Lemma 4.2 follows by setting $\delta = q^{-4n}$ and take a union bound for $\underline{\mathbf{A}}$. □

Lemma 4.3. *For $n', n, m, q \in \mathbb{N}$, $\sigma > 0$. $\overline{\mathbf{A}} \in \mathbb{Z}_q^{n' \times m}$, $\underline{\mathbf{A}} \in \mathbb{Z}_q^{n \times m}$. Assuming the columns of $\mathbf{A} := \binom{\overline{\mathbf{A}}}{\underline{\mathbf{A}}}$ generate $\mathbb{Z}_q^{n'+n}$. For any vectors $\mathbf{u} \in \mathbb{Z}_q^n$, $\mathbf{z} \in \mathbb{Z}_q^{n'}$, and $\mathbf{c} \in \mathbb{Z}^m$ where $\mathbf{A} \cdot \mathbf{c} = \binom{\mathbf{z}}{\mathbf{u}} \mod q$. The conditional distribution D of $\mathbf{x} \leftarrow \mathbf{c} + D_{\Lambda^\perp(\overline{\mathbf{A}}),\sigma,-\mathbf{c}}$ given $\underline{\mathbf{A}}\mathbf{x} = \mathbf{u} \mod q$ is exactly $\mathbf{c} + D_{\Lambda^\perp(\mathbf{A}),\sigma,-\mathbf{c}}$.*

Proof. Observe that the support of D is $\mathbf{c} + \Lambda^\perp(\mathbf{A})$. We compute the distribution D: for all $\mathbf{x} \in \mathbf{c} + \Lambda^\perp(\mathbf{A})$,

$$D(\mathbf{x}) = \frac{\rho_\sigma(\mathbf{x})}{\rho_\sigma(\mathbf{c} + \Lambda^\perp(\mathbf{A}))} = \frac{\rho_{\sigma,-\mathbf{c}}(\mathbf{x} - \mathbf{c})}{\rho_{\sigma,-\mathbf{c}}(\Lambda^\perp(\mathbf{A}))} = D_{\Lambda^\perp(\mathbf{A}),\sigma,-\mathbf{c}}(\mathbf{x} - \mathbf{c}). \tag{9}$$

□

Finally from Lemma 3.4, let $\Lambda = \Lambda^\perp(\overline{\mathbf{A}})$, $\Lambda' = \Lambda^\perp(\mathbf{A})$, we have $\Lambda' \subseteq \Lambda$. Since $\sigma > \eta_\epsilon(\Lambda')$, $D_{\Lambda^\perp(\overline{\mathbf{A}}),\sigma,-\mathbf{c}}$ is 2ϵ-statistically close to uniform over the cosets of the quotient group $(\Lambda^\perp(\overline{\mathbf{A}})/\Lambda^\perp(\mathbf{A}))$. The rest of the proof of Lemma 4.1 follows Lemma 4.3 and Lemma 4.2. □

4.2 The Computational Lemma

Lemma 4.4. *Given $n, m, k, q \in \mathbb{N}, \sigma \in \mathbb{R}$ such that $n, m, k \in \text{poly}(\lambda)$, $m \geq 4n\log q$, $\sigma \geq 2\sqrt{n\log q}$. For arbitrary matrix $\mathbf{Z} \in \mathbb{Z}_q^{n \times k}$, the following two distributions are computationally indistinguishable assuming $\mathsf{LWE}_{m,k,q,U(\mathbb{Z}_q),D_{\mathbb{Z},\sigma},D_{\mathbb{Z},\sigma}}$.*

Dist. 1 *Let $\mathbf{A}, \tau \leftarrow \mathsf{TrapSam}(1^n, 1^m, q)$, $\mathbf{E} \leftarrow D_{\mathbb{Z},\sigma}^{n \times k}$. Sample $\mathbf{D} \leftarrow \mathbf{A}^{-1}(\mathbf{Z} + \mathbf{E}, \sigma)$ using τ. Output $\mathbf{D}$.*

Dist. 2 *Sample* $\mathbf{D} = D_{\mathbb{Z},\sigma}^{m\times k}$. *Output* $\mathbf{D}$.

Proof. We prove a stronger statement where the computational indistinguishability holds even when $\mathbf{Z}$ is given to the adversary. The proof uses the Bonsai technique [CHKP12]. Let $m = m_1+m_2$ such that $m_1, m_2 \geq 2n\log q$. We introduce 2 intermediate distributions,

Dist. 1.1 Let $\mathbf{A}_1 \leftarrow U(\mathbb{Z}_q^{n\times m_1})$, $(\mathbf{A}_2, \tau_2) \leftarrow \mathsf{TrapSam}(1^n, 1^{m_2}, q)$. Sample $\mathbf{D}_1 \leftarrow D_{\mathbb{Z},\sigma}^{m_1\times k}$. Let $\mathbf{E} \leftarrow D_{\mathbb{Z},\sigma}^{n\times k}$, sample $\mathbf{D}_2 \leftarrow \mathbf{A}_2^{-1}((-\mathbf{A}_1\cdot\mathbf{D}_1 + \mathbf{E} + \mathbf{Z}), \sigma)$ using τ_2. Let $\mathbf{D} := \begin{pmatrix}\mathbf{D}_1\\ \mathbf{D}_2\end{pmatrix}$. Output $\mathbf{D}$.

Dist. 1.2 Let $\mathbf{A}_1 \leftarrow U(\mathbb{Z}_q^{n\times m_1})$, $(\mathbf{A}_2, \tau_2) \leftarrow \mathsf{TrapSam}(1^n, 1^{m_2}, q)$. Sample $\mathbf{D}_1 \leftarrow D_{\mathbb{Z},\sigma}^{m_1\times k}$. Let $\mathbf{U} \leftarrow U(\mathbb{Z}_q^{n\times k})$, sample $\mathbf{D}_2 \leftarrow \mathbf{A}_2^{-1}((\mathbf{U} + \mathbf{Z}), \sigma)$ using τ_2. Let $\mathbf{D} := \begin{pmatrix}\mathbf{D}_1\\ \mathbf{D}_2\end{pmatrix}$. Output $\mathbf{D}$.

Then Distributions 1 and 1.1 are statistically close following Lemma 3.13. Distributions 2 and 1.2 are statistically close following Lemma 3.12.

It remains to prove that Dist. 1.1 $\approx_c$ Dist. 1.2 assuming $\mathsf{LWE}_{m_1,k,q,U(\mathbb{Z}_q),D_{\mathbb{Z},\sigma},D_{\mathbb{Z},\sigma}}$. This follows by taking $(\mathbf{D}_1, -\mathbf{A}_1\cdot\mathbf{D}_1 + \mathbf{E})$ as the LWE sample, where $\mathbf{A}_1$ is the concatenation of n independent uniform secret vectors, $\mathbf{D}_1$ is the low-norm public matrix and $\mathbf{E}$ is the error matrix.

Formally, suppose there exists a p.p.t. distinguisher A for Dist. 1.1 and Dist. 1.2, we build a distinguisher A' for $\mathsf{LWE}_{m_1,k,q,U(\mathbb{Z}_q),D_{\mathbb{Z},\sigma},D_{\mathbb{Z},\sigma}}$. Given the challenge sample $(\mathbf{D}_1, \mathbf{Y}_1)$, A' runs $(\mathbf{A}_2, \tau_2) \leftarrow \mathsf{TrapSam}(1^n, 1^{m_2}, q)$, samples $\mathbf{D}_2 \leftarrow \mathbf{A}_2^{-1}((\mathbf{Y}_1 + \mathbf{Z}), \sigma)$ using τ_2, send $\mathbf{D} := \begin{pmatrix}\mathbf{D}_1\\ \mathbf{D}_2\end{pmatrix}$ to the adversary A. If A says it is from Dist. 1.1, then A' chooses "LWE"; if A says Dist. 1.2, then A' chooses "random". The success probability of A' is same to the success probability of A. □

5 Generalized GGH15 Encodings

We present the abstraction of *generalized GGH15 encodings*. The abstraction includes a construction framework and definitions of security notions.

5.1 The construction framework

We begin with a description of the construction:

Construction 5.1 (γ-GGH15 Encodings)**.** The randomized algorithm $\mathsf{ggh.encode}$ takes the following inputs

- Parameters[5] 1^λ, $h, n, m, q, t, w \in \mathbb{N}$, $\sigma \in \mathbb{R}^*$ and the description of a distribution χ over $\mathbb{Z}$.
- A function $\gamma : \mathbb{Z}^{w\times w} \times \mathbb{Z}^{n\times n} \to \mathbb{Z}^{t\times t}$.
- Matrices $\left\{\mathbf{M}_{i,b} \in \mathbb{Z}_{i,b}^{w\times w}\right\}_{i\in[h],b\in\{0,1\}}, \left\{\mathbf{S}_{i,b} \in \mathbb{Z}_{i,b}^{n\times n}\right\}_{i\in[h],b\in\{0,1\}}$.
- A matrix $\mathbf{A}_h \in \mathbb{Z}_q^{t\times m}$.

[5] In the rest of the presentation, these parameters are omitted in the input of $\mathsf{ggh.encode}$.

It generates the output as follows

- Samples $\left\{\mathbf{A}_i, \tau_i \leftarrow \mathsf{TrapSam}(1^t, 1^m, q)\right\}_{i\in\{0,1,\ldots,h-1\}}$.
- Samples $\left\{\mathbf{E}_{i,b} \leftarrow \chi^{t\times m}\right\}_{i\in[h],b\in\{0,1\}}$.
- For $i \in [h], b \in \{0,1\}$, let $\hat{\mathbf{S}}_{i,b} := \gamma(\mathbf{M}_{i,b}, \mathbf{S}_{i,b})$, then samples

$$\mathbf{D}_{i,b} \leftarrow \mathbf{A}_{i-1}^{-1}(\hat{\mathbf{S}}_{i,b} \cdot \mathbf{A}_i + \mathbf{E}_{i,b}, \sigma)$$

  using τ_{i-1}.
- Outputs $\mathbf{A}_0$, $\{\mathbf{D}_{i,b}\}_{i\in[h],b\in\{0,1\}}$.

We require γ to be multiplicatively homomorphic:

$$\gamma(\mathbf{M}, \mathbf{S}) \cdot \gamma(\mathbf{M}', \mathbf{S}') = \gamma(\mathbf{M} \cdot \mathbf{M}', \mathbf{S} \cdot \mathbf{S}')$$

Remark 5.2 (Comparison with GGH15)**.** The goal of the original GGH15 graded encodings in [GGH15] was to emulate the functionality provided by multi-linear maps with respect to some underlying directed acyclic graph. The basic unit of the construction is an encoding of a low-norm matrix $\hat{\mathbf{S}}$ along $\mathbf{A}_0 \mapsto \mathbf{A}_1$ given by $\mathbf{A}_0^{-1}(\hat{\mathbf{S}}\mathbf{A}_1 + \mathbf{E})$, where $\hat{\mathbf{S}}$ must be drawn from some high-entropy distribution to achieve any meaningful notion of security.

Following [CC17, GKW17b, GKW17a, WZ17], we think of $\hat{\mathbf{S}}$ as being deterministically derived from an arbitrary low-norm matrix $\mathbf{M}$ and a random low-norm matrix $\mathbf{S}$ via some fixed function γ given by $\gamma : (\mathbf{M}, \mathbf{S}) \mapsto \mathbf{M} \otimes \mathbf{S}$ in the afore-mentioned constructions. Here, we make γ an explicit parameter to the construction, so that we obtain a family of constructions parameterized by γ, which we refer to as the "γ-GGH15 encodings".

Looking ahead to Section 5.2, another advantage of decoupling $\hat{\mathbf{S}}$ into $\mathbf{M}$ and $\mathbf{S}$ is that we can now require semantic security for arbitrary inputs $\mathbf{M}$ and random choices of $\mathbf{S}$ (more precisely, arbitrary $\{\mathbf{M}_{i,b}\}_{i\in[h],b\in\{0,1\}}$ and random $\{\mathbf{S}_{i,b}\}_{i\in[h],b\in\{0,1\}}$), as considered in [WZ17]. Moreover, this notion of semantic security can be achieved under the LWE assumption for some specific γ and classes of matrices $\mathbf{M}$. Here, we make explicit the idea that semantic security should be defined with respect to some fixed auxiliary function aux of the matrices $\{\mathbf{S}_{i,b}\}_{i\in[h],b\in\{0,1\}}, \mathbf{A}_0, \ldots, \mathbf{A}_h$.

Functionality. The next lemma captures the functionality provided by the construction, namely that for all $\mathbf{x} \in \{0,1\}^h$,

$$\mathbf{A}_0 \cdot \mathbf{D}_{\mathbf{x}} \approx \gamma(\mathbf{M}_{\mathbf{x}}, \mathbf{S}_{\mathbf{x}}) \cdot \mathbf{A}_h$$

Lemma 5.3 (Functionality of γ-GGH15 encodings)**.** *Suppose γ is multiplicatively homomorphic. For all inputs to the Construction 5.1 s.t. $\sigma > \Omega(\sqrt{t \log q})$, $m > \Omega(t \log q)$, $\|\chi\| \le \sigma$; we have for all $\mathbf{x} \in \{0,1\}^h$, with all but negligible probability over the randomness in Construction 5.1,*

$$\|\mathbf{A}_0 \cdot \mathbf{D}_{\mathbf{x}} - \gamma(\mathbf{M}_{\mathbf{x}}, \mathbf{S}_{\mathbf{x}}) \cdot \mathbf{A}_h\|_\infty \le h \cdot \left(m\sigma \cdot \max_{i,b} \|\gamma(\mathbf{M}_{i,b}, \mathbf{S}_{i,b})\|\right)^h.$$

Proof. Recall $\hat{\mathbf{S}}_{i,b} = \gamma(\mathbf{M}_{i,b}, \mathbf{S}_{i,b})$. It is straight-forward to prove by induction that for all $h' = 0, 1, \ldots, h$:

$$\mathbf{A}_0 \cdot \prod_{k=1}^{h'} \mathbf{D}_{k,x_k} = \left(\prod_{i=1}^{h'} \hat{\mathbf{S}}_{i,x_i}\right) \mathbf{A}_{h'} + \sum_{j=1}^{h'} \left(\left(\prod_{i=1}^{j-1} \hat{\mathbf{S}}_{i,x_i}\right) \cdot \mathbf{E}_{j,x_j} \cdot \prod_{k=j+1}^{h} \mathbf{D}_{k,x_k} \right) \tag{10}$$

The base case $h' = 0$ holds trivially. The inductive step uses the fact that for all $h' = 1, \ldots, h$:

$$\mathbf{A}_{h'-1} \cdot \mathbf{D}_{h',x_{h'}} = \hat{\mathbf{S}}_{h',x_{h'}} \cdot \mathbf{A}_{h'} + \mathbf{E}_{h',x_{h'}}$$

From the homomorphic property of γ we can deduce that

$$\prod_{i=1}^{h} \hat{\mathbf{S}}_{i,x_i} = \prod_{i=1}^{h} \gamma(\mathbf{M}_{i,x_i}, \mathbf{S}_{i,x_i}) = \gamma(\mathbf{M}_{\mathbf{x}}, \mathbf{S}_{\mathbf{x}})$$

Finally, we bound the error term as follows:

$$\begin{aligned}
\|\mathbf{A}_0 \cdot \mathbf{D}_{\mathbf{x}} - \gamma(\mathbf{M}_{\mathbf{x}}, \mathbf{S}_{\mathbf{x}}) \cdot \mathbf{A}_h\|_\infty &= \left\| \sum_{j=1}^{h} \left(\prod_{i=1}^{j-1} (\hat{\mathbf{S}}_{i,x_i}) \cdot \mathbf{E}_{j,x_j} \cdot \prod_{k=j+1}^{h} \mathbf{D}_{k,x_k} \right) \right\|_\infty \\
&\leq h \cdot \sqrt{t} \cdot \sigma \cdot \left(\sqrt{t} \cdot \max_{i,b} \|\gamma(\mathbf{M}_{i,b}, \mathbf{S}_{i,b})\| \cdot \sigma \cdot \sqrt{m} \right)^{h-1} \\
&\leq h \cdot \left(\max_{i,b} \|\gamma(\mathbf{M}_{i,b}, \mathbf{S}_{i,b})\| \cdot \sigma \cdot m \right)^{h}
\end{aligned}$$

□

Looking ahead, in the applications we will set the parameters to ensure that the threshold $B := h \cdot (m\sigma \cdot \max_{i,b} \|\gamma(\mathbf{M}_{i,b}, \mathbf{S}_{i,b})\|)^h$ is relatively small compared to the modulus q.

Remark 5.4 (Dimensions of $\mathbf{A}_h$)**.** The construction and many analyses in this article can be obviously generalized to the cases where the dimensions of matrices are more flexible. As an example, the matrix $\mathbf{A}_h$ can be chosen from $\mathbb{Z}_q^t$ instead of $\mathbb{Z}_q^{t \times m}$ (as a result, $\mathbf{D}_{h,0}$, $\mathbf{D}_{h,1}$ are from $\mathbb{Z}^m$ instead of $\mathbb{Z}^{m \times m}$). This change maintains necessary functionalities, reduce the size of the construction, and is (more importantly) necessary for one of the proofs in the paper. For the ease of presentation we keep all the $\mathbf{A}$ matrices with the same dimension, all the $\mathbf{D}$ matrices with the same dimension, and mention the exceptions as they arise.

Interesting γ functions. We are interested in the following 3 γ functions:

- $\gamma_\otimes : \{0,1\}^{w \times w} \times \mathbb{Z}^{n \times n} \to \mathbb{Z}^{(wn) \times (wn)}$, $\mathbf{M}, \mathbf{S} \mapsto \mathbf{M} \otimes \mathbf{S}$.

  $\gamma_\otimes$ with permutation matrices $\mathbf{M}$ was introduced and studied in [CC17, GKW17b, GKW17a, WZ17].

- $\gamma_{\text{diag}} : \mathbb{Z}^{w \times w} \times \mathbb{Z}^{n \times n} \to \mathbb{Z}^{(w+n) \times (w+n)}$, $\mathbf{M}, \mathbf{S} \mapsto \begin{pmatrix} \mathbf{M} & \\ & \mathbf{S} \end{pmatrix}$.

  γ_{diag} is implicit in the constructions in [GGH+13b, HHSS17] and is central to the security analysis in this work.

- $\gamma_{\otimes\text{diag}} : \{0,1\}^{w\times w} \times \mathbb{Z}^{n\times n} \to \mathbb{Z}^{(wn+n)\times(wn+n)}$, $\mathbf{M}, \mathbf{S} \mapsto \begin{pmatrix} \mathbf{M}\otimes\mathbf{S} & \\ & \mathbf{S} \end{pmatrix}$.

  We introduce $\gamma_{\otimes\text{diag}}$ in this work, which would be central to the applications in this paper.

Note that all of the three γ functions are multiplicatively homomorphic and norm-preserving.

Remark 5.5 (Example for $\gamma_\otimes$ and relations with [GKW17b, GKW17a, WZ17])**.** As an example for $\gamma_\otimes$, take

$$\mathbf{M} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$

which is a permutation matrix corresponding to the cyclic shift π that sends $1 \mapsto 2 \mapsto 3 \mapsto 1$. Then, for any $\mathbf{S} \in \mathbb{Z}^{n\times n}$ and any $\mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \mathbf{A}^{(3)} \in \mathbb{Z}_q^{n\times m}$, we have:

$$(\mathbf{M}\otimes\mathbf{S}) \begin{pmatrix} \mathbf{A}^{(1)} \\ \mathbf{A}^{(2)} \\ \mathbf{A}^{(3)} \end{pmatrix} = \begin{pmatrix} \mathbf{0} & \mathbf{S} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{S} \\ \mathbf{S} & \mathbf{0} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{A}^{(1)} \\ \mathbf{A}^{(2)} \\ \mathbf{A}^{(3)} \end{pmatrix} = \begin{pmatrix} \mathbf{S}\mathbf{A}^{(2)} \\ \mathbf{S}\mathbf{A}^{(3)} \\ \mathbf{S}\mathbf{A}^{(1)} \end{pmatrix}$$

More generally, let $\pi : [w] \to [w]$ denote a permutation on $[w]$, and let $\mathbf{M} \in \{0,1\}^{w\times w}$ denote the standard representation of π as a matrix. Then, for any $\mathbf{S} \in \mathbb{Z}^{n\times n}$ and any $\mathbf{A} \in \mathbb{Z}_q^{nw\times m}$, we have

$$(\mathbf{M}\otimes\mathbf{S})\mathbf{A} = \begin{pmatrix} \mathbf{S}\mathbf{A}^{(\pi(1))} \\ \vdots \\ \mathbf{S}\mathbf{A}^{(\pi(w))} \end{pmatrix}, \text{where } \mathbf{A} = \begin{pmatrix} \mathbf{A}^{(1)} \\ \vdots \\ \mathbf{A}^{(w)} \end{pmatrix}, \mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(w)} \in \mathbb{Z}_q^{n\times m}$$

The latter is essentially how the $\gamma_\otimes$-GGH15 encoding is described using the notation in [GKW17b, GKW17a, WZ17].

5.2 Security notions

Intuitively, semantic security says that for all $\mathbf{M}$, the output of the γ-GGH15 encodings

$$\mathbf{A}_0, \{\mathbf{D}_{i,b}\}_{i\in[h],b\in\{0,1\}}$$

hides $\{\mathbf{M}_{i,b}\}_{i\in[h],b\in\{0,1\}}$, for random choices of $\{\mathbf{S}_{i,b}\}_{i\in[h],b\in\{0,1\}}$ and $\mathbf{A}_0, \ldots, \mathbf{A}_h$. We consider a more general notion parameterized by some fixed function aux of $\{\mathbf{S}_{i,b}\}_{i\in[h],b\in\{0,1\}}, \mathbf{A}_0, \ldots, \mathbf{A}_h$, and we require that $\mathsf{aux}, \{\mathbf{D}_{i,b}\}_{i\in[h],b\in\{0,1\}}$ hides $\{\mathbf{M}_{i,b}\}_{i\in[h],b\in\{0,1\}}$.

Definition 3 (Semantic security with auxiliary input)**.** We say that the γ-GGH15 encodings satisfies semantic security with auxiliary input aux for a family of matrices $\mathcal{M} \subseteq \mathbb{Z}^{w\times w}$ if for all $\{\mathbf{M}_{i,b} \in \mathcal{M}\}_{i\in[h],b\in\{0,1\}}$, we have

$$\mathsf{aux}, \{\mathbf{D}_{i,b}\}_{i\in[h],b\in\{0,1\}} \approx_c \mathsf{aux}, \left\{(D_{\mathbb{Z},\sigma}^{m\times m})_{i,b}\right\}_{i\in[h],b\in\{0,1\}}$$

where

$$\mathbf{S}_{i,b} \leftarrow D_{\mathbb{Z},\sigma}^{n\times n}, \mathbf{A}_h \leftarrow U(\mathbb{Z}_q^{t\times m}), \{\mathbf{D}_{i,b}\} \leftarrow \mathsf{ggh.encode}(\gamma, \{\mathbf{M}_{i,b}\}_{i\in[h],b\in\{0,1\}}, \{\mathbf{S}_{i,b}\}_{i\in[h],b\in\{0,1\}}, \mathbf{A}_h)$$

and aux is a fixed function of $\{\mathbf{S}_{i,b}\}_{i\in[h],b\in\{0,1\}}, \mathbf{A}_0, \ldots, \mathbf{A}_h$.

Remark 5.6 ($\gamma_\otimes$-GGH encodings with permutation matrices)**.** Canetti and Chen [CC17] (also, [GKW17a, WZ17]) showed that the $\gamma_\otimes$-GGH15 encoding satisfies semantic security with auxiliary input $(\mathbf{A}_0, \mathbf{A}_1, \ldots, \mathbf{A}_h)$ for the family of permutation matrices in $\{0,1\}^{w\times w}$.

We can prove that the $\gamma_\otimes$-GGH15 encoding satisfies semantic security with auxiliary input $(\mathbf{A}_0, \{\mathbf{S}_{i,b}\}_{i\in[\ell],b\in\{0,1\}})$ for the family of permutation matrices in $\{0,1\}^{w\times w}$, by using the LWE assumption with the $\mathbf{S}_{i,b}$ as the public matrices. Such a proof requires a multiplicative blow-up (of roughly $O(\log q)$) in the dimensions of the $\mathbf{S}_{i,b}$ matrices. One of the advantages of using the $\mathbf{S}$ matrices as the public matrices is that we can use the same $\mathbf{S}_0, \mathbf{S}_1$ across all the h levels, similar to the PRF construction in [BLMR13].

5.3 Semantic security for γ_{diag}-GGH15 and $\gamma_{\otimes\text{diag}}$-GGH15 encodings

In this section, we prove semantic security of the γ_{diag}-GGH15 and $\gamma_{\otimes\text{diag}}$-GGH15 encodings in Construction 5.1 under the LWE assumption, where

$$\gamma_{\text{diag}}(\mathbf{M},\mathbf{S}) = \begin{pmatrix}\mathbf{M} & \\ & \mathbf{S}\end{pmatrix},\ \gamma_{\otimes\text{diag}}(\mathbf{M},\mathbf{S}) = \begin{pmatrix}\mathbf{M}\otimes\mathbf{S} & \\ & \mathbf{S}\end{pmatrix}.$$

In fact, we show that this holds given auxiliary input about $\mathbf{A}_0$ and $\{\mathbf{S}_{i,b}\}_{i\in[h],b\in\{0,1\}}$.

S-dependent security. Concretely, we will derive semantic security of $\gamma_{\otimes\text{diag}}$ from that of γ_{diag} by showing that the construction γ_{diag} satisfies a stronger notion of $\mathbf{S}$-dependent security where the matrices $\{\mathbf{M}_{i,b}\}_{i\in[h],b\in\{0,1\}}$ may depend on $\{\mathbf{S}_{i,b}\}_{i\in[h],b\in\{0,1\}}$:

Definition 4 ($\mathbf{S}$-dependent semantic security with auxiliary input)**.** We say that the γ-GGH15 encodings satisfies $\mathbf{S}$-dependent semantic security with auxiliary input aux for a family of matrices $\mathcal{M}\subseteq\mathbb{Z}^{w\times w}$ if for every polynomial-size circuit $f:(\mathbb{Z}^{n\times n})^{2h}\to\mathcal{M}^{2h}$, we have

$$\mathsf{aux}, \{\mathbf{D}_{i,b}\}_{i\in[h],b\in\{0,1\}} \approx_c \mathsf{aux}, \left\{(D_{\mathbb{Z},\sigma}^{m\times m})_{i,b}\right\}_{i\in[h],b\in\{0,1\}}$$

where

$$\begin{aligned}\mathbf{S}_{i,b}\leftarrow D_{\mathbb{Z},\sigma}^{n\times n}, \mathbf{A}_h\leftarrow U(\mathbb{Z}_q^{t\times m}), \{\mathbf{M}_{i,b}\}_{i\in[h],b\in\{0,1\}} = f(\{\mathbf{S}_{i,b}\}_{i\in[h],b\in\{0,1\}}),\\ \{\mathbf{D}_{i,b}\}\leftarrow \mathsf{ggh.encode}(\gamma, \{\mathbf{S}_{i,b}\}_{i\in[h],b\in\{0,1\}}, \{\mathbf{M}_{i,b}\}_{i\in[h],b\in\{0,1\}}, \mathbf{A}_h)\end{aligned}$$

and aux is a fixed function of $\{\mathbf{S}_{i,b}\}_{i\in[h],b\in\{0,1\}}, \mathbf{A}_0, \ldots, \mathbf{A}_h$.

Theorem 5.7 ($\mathbf{S}$-dependent semantic security of γ_{diag})**.** *Assuming* $\mathsf{LWE}_{n,2m,q,U(\mathbb{Z}_q),D_{\mathbb{Z},\sigma},D_{\mathbb{Z},\sigma}}$*, the* γ_{diag}*-GGH15 encodings in Construction 5.1 satisfies* $\mathbf{S}$*-dependent semantic security for* $\mathcal{M}=\mathbb{Z}^{w\times w}$ *with auxiliary input*

$$\mathsf{aux} = \{\mathbf{S}_{i,b}\}_{i\in[h],b\in\{0,1\}}, \mathbf{J}\cdot\mathbf{A}_0, \overline{\mathbf{A}}_h$$

where $\overline{\mathbf{A}}_h\in\mathbb{Z}_q^{w\times m}$ *is the top* w *rows of* $\mathbf{A}_h$ *and* $\mathbf{J}\in\{0,1\}^{n\times(t-n)} \mid \mathbf{I}^{n\times n}$.

Remark 5.8 (Necessity of $\mathbf{JA}_0$)**.** Ideally, we would liked to have shown that semantic security holds with auxiliary input $\mathbf{A}_0$ (as opposed to $\mathbf{JA}_0$). However, such a statement is false for general $\mathcal{M}\in\mathbb{Z}^{w\times w}$. Concretely, given $\mathbf{A}_0, \mathbf{D}_{1,0}$, we can compute $\mathbf{A}_0\cdot\mathbf{D}_{1,0}$ which leaks information about the structure of $\mathbf{M}_{1,0}$. In particular, we can distinguish between $\begin{pmatrix}1&0\\0&1\end{pmatrix}$ and $\begin{pmatrix}0&1\\0&1\end{pmatrix}$.

As an immediate corollary, we then have:

Corollary 5.9 (semantic security of $\gamma_{\otimes\text{diag}}$)**.** *Assuming* $\mathsf{LWE}_{n,2m,q,U(\mathbb{Z}_q),D_{\mathbb{Z},\sigma},D_{\mathbb{Z},\sigma}}$*, the* $\gamma_{\otimes diag}$*-GGH15 encodings in Construction 5.1 satisfies semantic security for* $\mathcal{M} = \mathbb{Z}^{w\times w}$ *with auxiliary input*

$$\mathsf{aux} = \{\mathbf{S}_{i,b}\}_{i\in[h],b\in\{0,1\}}, \mathbf{J}\cdot\mathbf{A}_0, \overline{\mathbf{A}}_h$$

where $\overline{\mathbf{A}}_h \in \mathbb{Z}_q^{wn\times m}$ *is the top* wn *rows of* $\mathbf{A}_h$ *and* $\mathbf{J} \in \{0,1\}^{n\times(t-n)} \mid \mathbf{I}^{n\times n}$*.*

5.4 Proof of the main theorem

Proof of Theorem 5.7. For $t, n, w \in \mathbb{N}$ such that $t = w + n$. For any matrix $\mathbf{X} \in \mathbb{Z}^{t\times *}$, let $\mathbf{X} = \begin{pmatrix}\overline{\mathbf{X}}\\ \underline{\mathbf{X}}\end{pmatrix}$, where $\overline{\mathbf{X}} \in \mathbb{Z}^{w\times *}$, $\underline{\mathbf{X}} \in \mathbb{Z}^{n\times *}$. For the sake of completeness we spell out the details of the real and simulated distributions which will be proven indistinguishable.

The real and simulated distributions. In the real distribution the adversary is given

$$\mathbf{J}\cdot\mathbf{A}_0, \left\{\boxed{\mathbf{D}_{i,b}}, \mathbf{S}_{i,b}, \mathbf{M}_{i,b}\right\}_{i\in[h],b\in\{0,1\}}, \overline{\mathbf{A}}_h$$

where

- $\left\{\mathbf{A}_i, \tau_i \leftarrow \mathsf{TrapSam}(1^t, 1^m, q)\right\}_{i\in\{0,1,\ldots,h-1\}}, \mathbf{A}_h \leftarrow U(\mathbb{Z}_q^{t\times m})$
- $\mathbf{S}_{i,b} \leftarrow D_{\mathbb{Z},\sigma}^{n\times n}, \{\mathbf{M}_{i,b}\}_{i\in[h],b\in\{0,1\}} \leftarrow f(\{\mathbf{S}_{i,b}\}_{i\in[h],b\in\{0,1\}})$
- $\mathbf{D}_{i,b} \leftarrow \mathbf{A}_{i-1}^{-1}\begin{pmatrix}\mathbf{M}_{i,b}\overline{\mathbf{A}}_i + \overline{\mathbf{E}}_{i,b}\\ \mathbf{S}_{i,b}\underline{\mathbf{A}}_i + \underline{\mathbf{E}}_{i,b}\end{pmatrix}, \mathbf{E}_{i,b} \leftarrow \chi^{t\times m}$

The simulated distribution is generated in the same way except that the adversary is given

$$\mathbf{J}\cdot\mathbf{A}_0, \left\{\boxed{\mathbf{V}_{i,b}}, \mathbf{S}_{i,b}, \mathbf{M}_{i,b}\right\}_{i\in[h],b\in\{0,1\}}, \overline{\mathbf{A}}_h$$

where $\mathbf{V}_{i,b} \leftarrow D_{\mathbb{Z},\sigma}^{m\times m}$.

To show that the real distribution is computationally indistinguishable from the simulated one, we introduce the following intermediate distributions.

Distributions 1.i, for $i \in \{h+1, h, ..., 1\}$**.** Let Distribution $1.(h+1)$ be identical to the real distribution. For $i = h$ down to 1, let Distributions $1.i$ be the same to Distributions $1.(i+1)$, except that $\mathbf{A}_{i-1}$, $\mathbf{D}_{i,0}$, $\mathbf{D}_{i,1}$ are sampled differently. Let $(\overline{\mathbf{A}}_{i-1}, \tau_{i-1}) \leftarrow \mathsf{TrapSam}(1^w, 1^m, q)$, $\underline{\mathbf{A}}_{i-1} \leftarrow U(\mathbb{Z}_q^{n\times m})$. Sample $\mathbf{D}_{i,b} \leftarrow \overline{\mathbf{A}}_{i-1}^{-1}((\mathbf{M}_{i,b}\overline{\mathbf{A}}_i + \overline{\mathbf{E}}_{i,b}), \sigma)$ using τ_{i-1}, $b \in \{0,1\}$.

Distributions 2.0. Distribution 2.0 is sampled identically to Distribution 1.1, except that $\mathbf{J}\cdot\mathbf{A}_0$ is replaced with a uniformly random matrix $\mathbf{U} \xleftarrow{\$} \mathbb{Z}^{n\times m}$. Since $\mathbf{J} \in \{0,1\}^{n\times(t-n)} \mid \mathbf{I}^{n\times n}$, $\mathbf{U} \approx_s \mathbf{J}\cdot\mathbf{A}_0$ for $\mathbf{A}_0, \tau_0 \leftarrow \mathsf{TrapSam}(1^t, 1^m, q)$ due to Lemma 3.10.

Distributions 2.j, for $j \in \{1, ..., h\}$. For $j = 1, 2, ..., h$, let Distributions $2.j$ be the same to Distributions $2.(j-1)$, except that $\mathbf{D}_{j,0}$, $\mathbf{D}_{j,1}$ are sampled simply from $D_{\mathbb{Z},\sigma}^{m\times m}$. Note that Dist. $2.h$ is identical to the simulated distribution, except that in Dist. $2.h$, $\mathbf{U} \xleftarrow{\$} \mathbb{Z}^{n\times m}$ is in the place where $\mathbf{J} \cdot \mathbf{A}_0$ is in the simulated distribution, so they are statistically close again due to Lemma 3.10.

The sequence. We will show that:

$$\text{Real} = 1.(h+1) \approx_c 1.h \approx_c \cdots \approx_c 1.1 \approx_s 2.0 \approx_c 2.1 \approx_c \cdots \approx_c 2.h \approx_s \text{Simulated}$$

In particular, the $\approx_c$'s will rely on the LWE assumption, using $\mathbf{A}_1, \ldots, \mathbf{A}_\ell$ as LWE secrets in the following order: $\underline{\mathbf{A}}_\ell, \ldots, \underline{\mathbf{A}}_1, \overline{\mathbf{A}}_0, \ldots, \overline{\mathbf{A}}_{\ell-1}$.

Lemma 5.10. *For $i \in [h]$, Distribution $1.(i+1) \approx_c$ Distribution $1.i$ assuming $\mathsf{LWE}_{n,2n,q,U(\mathbb{Z}_q),D_{\mathbb{Z},\sigma},D_{\mathbb{Z},\sigma}}$.*

Roughly speaking, we will show that for all $i \in [h]$,

$$\left\{ \mathbf{A}_{i-1}^{-1} \begin{pmatrix} \mathbf{M}_{i,b}\overline{\mathbf{A}}_i + \overline{\mathbf{E}}_{i,b} \\ \mathbf{S}_{i,b}\underline{\mathbf{A}}_i + \underline{\mathbf{E}}_{i,b} \end{pmatrix} \right\}_{b\in\{0,1\}} \approx_c \left\{ \overline{\mathbf{A}}_{i-1}^{-1}(\mathbf{M}_{i,b}\overline{\mathbf{A}}_i + \overline{\mathbf{E}}_{i,b}) \right\}_{b\in\{0,1\}}$$

where the distinguisher is also given $\mathbf{A}_{i-1}, \tau_{i-1}, \mathbf{S}_{i,0}, \mathbf{S}_{i,1}, \mathbf{M}_{i,0}, \mathbf{M}_{i,1}, \overline{\mathbf{A}}_i$, but not $\underline{\mathbf{A}}_i$, so that we can treat $\underline{\mathbf{A}}_i$ as a LWE secret, cf. Lemma 4.4.

Proof. We introduce an intermediate distribution $1.i^*$, which is generated in the same way as Distributions $1.(i+1)$, except that $\mathbf{D}_{i,0}$, $\mathbf{D}_{i,1}$ are sampled as:

$$\mathbf{D}_{i,b} \leftarrow \mathbf{A}_{i-1}^{-1} \left(\begin{pmatrix} \mathbf{M}_{i,b}\overline{\mathbf{A}}_i + \overline{\mathbf{E}}_{i,b} \\ \mathbf{U}_{i,b} \end{pmatrix}, \sigma \right), b \in \{0,1\}.$$

where $(\mathbf{U}_{i,0}, \mathbf{U}_{i,1}) \leftarrow U(\mathbb{Z}_q^{n\times m} \times \mathbb{Z}_q^{n\times m})$.

The intermediate distribution $1.i^*$ is statistically close to Distribution $1.i$ due to Lemma 4.1. It remains to prove that $1.i^*$ is computationally indistinguishable from Distribution $1.(i+1)$. This follows Lemma 3.9, by treating $\underline{\mathbf{A}}_i$ as the LWE secret, and $\mathbf{S}_{i,0}, \mathbf{S}_{i,1}$ as the public matrices.

Formally, if there's an adversary A that distinguishes Distributions $1.(i+1)$ and $1.i^*$, we build a distinguisher A' for $\mathsf{LWE}_{n,2n,q,U(\mathbb{Z}_q),D_{\mathbb{Z},\sigma},D_{\mathbb{Z},\sigma}}$ as follows. Once given the LWE challenge

$$\mathbf{S}_{i,0}, \mathbf{S}_{i,1}, \underline{\mathbf{Y}}_{i,0}, \underline{\mathbf{Y}}_{i,1}$$

where $\mathbf{S}_{i,0}, \mathbf{S}_{i,1}$ are the low-norm public matrices, $\underline{\mathbf{Y}}_{i,0}, \underline{\mathbf{Y}}_{i,1}$ are either the $\mathsf{LWE}_{n,2n,q,U(\mathbb{Z}_q),D_{\mathbb{Z},\sigma},D_{\mathbb{Z},\sigma}}$ samples with the common secret $\underline{\mathbf{A}}_i \leftarrow U(\mathbb{Z}_q^{n\times m})$, or independent uniform samples from $\mathbb{Z}_q^{n\times m} \times \mathbb{Z}_q^{n\times m}$. The LWE distinguisher A' proceeds as follows:

1. Sample $\left\{ \mathbf{S}_{k,b} \leftarrow D_{\mathbb{Z},\sigma}^{n\times n} \right\}_{k\in[h],k\neq i,b\in\{0,1\}}$.
2. For $k \in [h], b \in \{0,1\}$, compute $\mathbf{M}_{k,b} \in \mathbb{Z}^{w\times w}$ using $f(\{\mathbf{S}_{k,b}\}_{k\in[h],b\in\{0,1\}})$.
3. For $k \in \{0, 1, ..., i-1\}$, sample $\mathbf{A}_k, \tau_k \leftarrow \mathsf{TrapSam}(1^t, 1^m, q)$. For $k \in \{i, i+1, ..., h-1\}$, sample $\overline{\mathbf{A}}_k, \bar{\tau}_k \leftarrow \mathsf{TrapSam}(1^w, 1^m, q)$. Sample $\overline{\mathbf{A}}_h \leftarrow U(\mathbb{Z}_q^{t\times m})$.

4. For $k \in [h], b \in \{0,1\}$, samples

$$\mathbf{D}_{k,b} \leftarrow \begin{cases} \mathbf{A}_{k-1}^{-1}\begin{pmatrix}\mathbf{M}_{k,b}\overline{\mathbf{A}}_k+\overline{\mathbf{E}}_{k,b}\\ \mathbf{S}_{k,b}\underline{\mathbf{A}}_k+\underline{\mathbf{E}}_{k,b}\end{pmatrix} & \text{using } \tau_{k-1} \text{ if } k \le i-1 \\ \mathbf{A}_{i-1}^{-1}\begin{pmatrix}\mathbf{M}_{i,b}\overline{\mathbf{A}}_i+\overline{\mathbf{E}}_{i,b}\\ \underline{\mathbf{Y}}_{i,b}\end{pmatrix} & \text{using } \tau_{i-1} \text{ if } k = i \\ \overline{\mathbf{A}}_{k-1}^{-1}(\mathbf{M}_{k,b}\overline{\mathbf{A}}_k + \overline{\mathbf{E}}_{k,b})) & \text{using } \bar{\tau}_{k-1} \text{ if } k \ge i+1 \end{cases}$$

with standard deviation σ.

The LWE distinguisher A' then sends

$$\mathbf{J}\cdot\mathbf{A}_0, \left\{\boxed{\mathbf{D}_{k,b}}, \mathbf{S}_{k,b}, \mathbf{M}_{k,b}\right\}_{k\in[h],b\in\{0,1\}}, \overline{\mathbf{A}}_h.$$

to the adversary A. If A says it is Dist. $1.(i+1)$, it corresponds to the LWE samples with low-norm public matrices; if A says Dist. $1.i^*$, it corresponds to the uniform distribution. □

Lemma 5.11. *For $j \in [h]$, Distribution $2.(j-1) \approx_c$ Distributions $2.j$ assuming* $\mathsf{LWE}_{m,2m,q,U(\mathbb{Z}_q),D_{\mathbb{Z},\sigma},D_{\mathbb{Z},\sigma}}$.

Roughly speaking, we will show that for all $j \in [h]$,

$$\left\{\overline{\mathbf{A}}_{j-1}^{-1}(\mathbf{M}_{j,b}\overline{\mathbf{A}}_j + \overline{\mathbf{E}}_{j,b})\right\}_{b\in\{0,1\}} \approx_c \left\{D_{\mathbb{Z},\sigma}^{m\times m}\right\}_{b\in\{0,1\}}$$

where the distinguisher is also given $\mathbf{M}_{j,0}, \mathbf{M}_{j,1}, \overline{\mathbf{A}}_j$, but not $\overline{\mathbf{A}}_{j-1}$, so as to trigger Lemma 4.4.

Proof. For $j \in [h]$, suppose there exists an adversary A that distinguishes Distributions $2.(j-1)$ and $2.j$, we build a distinguisher A' for Distributions 1 and 2 in Lemma 4.4 as follows. Given challenging samples

$$\mathbf{D}_{j,0} \mid \mathbf{D}_{j,1} \in \mathbb{Z}^{m\times 2m}$$

either obtained from $\overline{\mathbf{A}}_{j-1}^{-1}(\left[\mathbf{M}_{j,0}\overline{\mathbf{A}}_j + \overline{\mathbf{E}}_{j,0} \mid \mathbf{M}_{j,1}\overline{\mathbf{A}}_j + \overline{\mathbf{E}}_{j,1}\right])$ which corresponds to Dist. 1 in Lemma 4.4 (by treating $\left[\mathbf{M}_{j,0}\overline{\mathbf{A}}_j \mid \mathbf{M}_{j,1}\overline{\mathbf{A}}_j\right]$ as the arbitrary matrix $\mathbf{Z}$); or from $D_{\mathbb{Z},\sigma}^{m\times 2m}$ which corresponds to Dist. 2 in Lemma 4.4. The distinguisher A' proceeds as follows:

1. For $k \in [h], b \in \{0,1\}$, sample $\mathbf{S}_{k,b} \leftarrow D_{\mathbb{Z},\sigma}^{n\times n}$.
2. For $k \in [h], b \in \{0,1\}$, compute $\mathbf{M}_{k,b} \in \mathbb{Z}^{w\times w}$ using $f(\{\mathbf{S}_{k,b}\}_{k\in[h],b\in\{0,1\}})$.
3. For $k \in \{j, j+1, ..., h-1\}$, sample $\overline{\mathbf{A}}_k, \bar{\tau}_k \leftarrow \mathsf{TrapSam}(1^w, 1^m, q)$. Sample $\overline{\mathbf{A}}_h \leftarrow U(\mathbb{Z}_q^{t\times m})$.
4. For $k \in \{1, 2, ..., j-1, j+1, ..., h\}, b \in \{0,1\}$, samples

$$\mathbf{D}_{k,b} \leftarrow \begin{cases} D_{\mathbb{Z},\sigma}^{m\times m} \text{ if } k \le j-1 \\ \overline{\mathbf{A}}_{k-1}^{-1}(\mathbf{M}_{k,b}\overline{\mathbf{A}}_k + \overline{\mathbf{E}}_{k,b}, \sigma) \text{ using } \bar{\tau}_{k-1} \text{ if } k \ge j+1 \end{cases}.$$

5. Sample $\mathbf{U} \leftarrow U(\mathbb{Z}_q^{n\times m})$.

A' then sends

$$\mathbf{U}, \left\{\boxed{\mathbf{D}_{k,b}}, \mathbf{S}_{k,b}, \mathbf{M}_{k,b}\right\}_{k\in[h],b\in\{0,1\}}, \overline{\mathbf{A}}_h.$$

to the adversary A. Note that A' correctly produce the output without $\overline{\mathbf{A}}_{j-1}$. So if A determines that the samples are from Distribution $2.(j-1)$, A' chooses Dist. 1 in Lemma 4.4; if A determines that the samples are from Distribution $2.j$, A' chooses Dist. 2 in Lemma 4.4. □

Theorem 5.7 follows from Lemmas 5.10 and 5.11. □

6 Matrix branching programs

To formally describe the applications it is helpful to introduce the terminologies for *matrix branching programs*.

Definition 5 (Matrix branching program)**.** Let $\ell, h \in \mathbb{N}$ be the bit-length of the input and the index of a branching program. An index-to-input map $\iota : [h] \to [\ell]$ and an input-to-index map $\varpi : \{0,1\}^\ell \to \{0,1\}^h$ come in pairs, i.e. $\varpi(\mathbf{x})_i = x_{\iota(i)}, \forall i \in [h], \mathbf{x} \in \{0,1\}^\ell$.

A dimension-w, length-h matrix branching program over ℓ-bit inputs consists of a pair of maps (ϖ, ι), a sequence of pairs of 0-1 matrices, and two disjoint sets of target matrices

$$\Gamma = \left\{\varpi, \iota,\ \left\{\mathbf{M}_{i,b} \in \{0,1\}^{w\times w}\right\}_{i\in[h], b\in\{0,1\}}, \mathcal{P}_0, \mathcal{P}_1 \subset \{0,1\}^{w\times w}\right\}.$$

This branching program is computing the function $f_\Gamma : \{0,1\}^\ell \to \{0,1\}$, defined as

$$f_\Gamma(x) = \begin{cases} 0 & \text{if } \mathbf{M}_{\varpi(\mathbf{x})} = \prod_{i\in[h]} \mathbf{M}_{i,x_{\iota(i)}} \in \mathcal{P}_0 \\ 1 & \text{if } \mathbf{M}_{\varpi(\mathbf{x})} = \prod_{i\in[h]} \mathbf{M}_{i,x_{\iota(i)}} \in \mathcal{P}_1 \end{cases}$$

Remark 6.1. Since one of ι, ϖ uniquely defines the other, in the branching programs defined in this paper we typically specify only one of them.

Looking ahead, the applications in this paper may require additional feature on the target sets $\mathcal{P}_0, \mathcal{P}_1$ to perform the correct functionality. The following 2 types suffice for most of the applications in this paper (the exceptions are otherwise defined locally).

Definition 6 (Type I branching programs)**.** For a vector $\mathbf{v} \in \{0,1\}^{1\times w}$, an input-to-index map ϖ, an integer w. The set of Type I branching programs $\mathcal{G}_{\mathbf{v},\varpi,w}$ satisfies

1. All $\Gamma \in \mathcal{G}_{\mathbf{v},\varpi,w}$ are associated with the same input-to-index map ϖ; all the matrices in Γ have the same dimension w.
2. For all $\Gamma \in \mathcal{G}_{\mathbf{v},\varpi,w}$, the target sets $\mathcal{P}_0, \mathcal{P}_1$ satisfies $\mathbf{v}\cdot\mathcal{P}_1 = \left\{\mathbf{0}^{1\times w}\right\}$; $\mathbf{v}\cdot\mathcal{P}_0 \subseteq \{0,1\}^{1\times w} \setminus \left\{\mathbf{0}^{1\times w}\right\}$.

Definition 7 (Type II branching programs)**.** For a vector $\mathbf{v} \in \{0,1\}^{1\times w}$, an input-to-index map ϖ, an integer $w \geq 2$. The set of Type II matrix branching programs $\mathcal{G}_{\mathbf{v},\varpi,w}$ satisfies

1. All $\Gamma \in \mathcal{G}_{\mathbf{v},\varpi,w}$ are associated with the same input-to-index map ϖ; all the matrices in Γ have the same dimension w.
2. For all $\Gamma \in \mathcal{G}_{\mathbf{v},\varpi,w}$, the target sets $\mathcal{P}_0$, $\mathcal{P}_1$ satisfies $\mathbf{v}\cdot\mathcal{P}_1 = \{\mathbf{e}_1\}$; $\mathbf{v}\cdot\mathcal{P}_0 = \{\mathbf{e}_2\}$, where $\mathbf{e}_i \in \{0,1\}^{1\times w}$ denotes the unit vector with the i^{th} coordinate being 1, the rest being 0.

Note that we can always transform Type II branching program into a Type I branching program by doubling the width while preserving the length:

Claim 6.2 (Type II $\subseteq$ Type I)**.** $\mathcal{G}^{\text{II}}_{\mathbf{v},\varpi,w} \subseteq \mathcal{G}^{\text{I}}_{\mathbf{v}|-\mathbf{v},\varpi,2w}$

The idea is to simply add a dummy branch that always computes some fixed matrix in $\mathcal{P}_1$; this doubles the width of the branching program. This transformation is used implicitly in several iO candidates, e.g. [GGH+13b, GGH15, HHSS17].

The branching programs obtained by Barrington's theorem directly satisfy Definition 7.

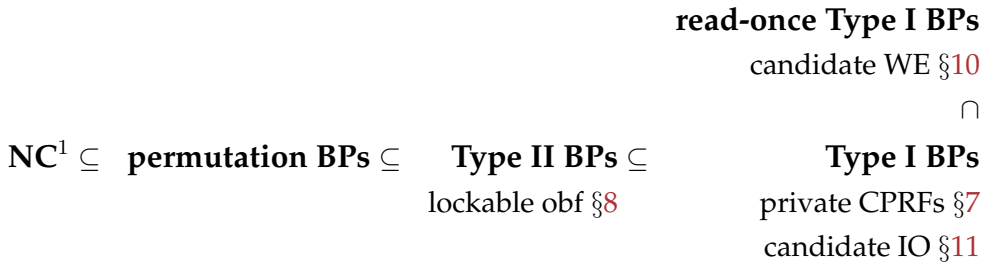

Figure 1: Relationships between different types of BPs and how they relate to our constructions.

Theorem 6.3 (Barrington's theorem [Bar86])**.** *For $d, w \in \mathbb{N}$, $w \geq 5$, and for any set of depth-d fan-in-2 Boolean circuits, there is a set of width-w length-4^d Type II branching programs, where each Γ is associated with the same index-to-input map ι, the same target matrices $\mathbf{P}_0$, $\mathbf{P}_1$, and permutation matrices $\{\mathbf{M}_{i,b} \in \{0,1\}^{w \times w}\}_{i \in [4^d], b \in \{0,1\}}$.*

The relationships between the different kinds of BPs and how they relate to our constructions is summarized in Fig. 1.

6.1 Representing CNFs as matrix branching programs

We describe two specific branching program representations of CNFs which will be used in our constructions. In both constructions, we obtain read-once branching programs of length ℓ for ℓ-bit inputs, whereas if we are restricted to permutations, then the known constructions require length at least ℓ^2 or width 2^ℓ.

The first one is from [GLW14]. The resulting branching programs are read-once Type I, and the matrices are diagonal.

Construction 6.4 (Type I BP representation of CNF)**.** Given a conjunction normal form (CNF) formula Ψ with ℓ variables and w clauses, namely $\Psi = \psi_1 \wedge \ldots \wedge \psi_w$, where each clause ψ_i is a disjunction of the literals $x_1, ..., x_\ell$ and their negations $\bar{x}_1, ..., \bar{x}_\ell$.

Construct $\Gamma_\Psi = \left\{\iota, \{\mathbf{M}_{i,b} \in \{0,1\}^{w \times w}\}_{i \in [\ell], b \in \{0,1\}}, \mathcal{P}_0, \mathcal{P}_1\right\}$ such that $\iota(v) = v, v \in [\ell]$; $\mathcal{P}_1 = \{\mathbf{0}^{w \times w}\}$, $\mathcal{P}_0$ is the set of w-dimensional 0-1 diagonal matrices excluding $\mathbf{0}^{w \times w}$. In other words,

$$\begin{cases} \mathbf{M}_\mathbf{x} = \mathbf{0} & \text{if } \Psi(\mathbf{x}) = 1 \\ \mathbf{M}_\mathbf{x} \neq \mathbf{0} & \text{if } \Psi(\mathbf{x}) = 0 \end{cases}.$$

The matrices in Γ_Ψ are constructed as follows.

1. Initialization: for all $i \in [\ell], b \in \{0,1\}$, Let $\mathbf{M}_{i,b} := \mathbf{I}^{w \times w}$.
2. If x_i appears in ψ_j: set the j^{th} entry on the diagonal of $\mathbf{M}_{i,1}$ to be 0.
3. If $\bar{x}_i$ appears in ψ_j: set the j^{th} entry on the diagonal of $\mathbf{M}_{i,0}$ to be 0.

This branching program has length ℓ and width w.

The second representation handles a single clause formed by a disjunction of the literals (we assume an empty clause always outputs 0; a clause that always outputs 1 can be simply represented by $x_1 \vee \bar{x}_1$). It belongs to Type II.

Construction 6.5 (Type II BP representation of a disjunction)**.** Given a clause ψ which is a disjunction of the literals $x_1, ..., x_\ell$ and their negations $\bar{x}_1, ..., \bar{x}_\ell$.

Let $\mathbf{N} = \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix}$. Construct $\Gamma_\psi = \left\{\iota, \left\{\mathbf{M}_{i,b} \in \{0,1\}^{2\times 2}\right\}_{i\in[\ell],b\in\{0,1\}}, \mathcal{P}_0, \mathcal{P}_1\right\}$ such that $\iota(v) = v, v \in [\ell]$; $\mathcal{P}_0 = \left\{\mathbf{I}^{2\times 2}\right\}$, $\mathcal{P}_1 = \{\mathbf{N}\}$. The matrices in Γ_ψ are constructed as follows.

1. Initialization: for all $i \in [\ell], b \in \{0,1\}$, Let $\mathbf{M}_{i,b} := \mathbf{I}^{2\times 2}$.
2. If x_i appears in ψ, let $\mathbf{M}_{i,1} = \mathbf{N}$.
3. If $\bar{x}_i$ appears in ψ, let $\mathbf{M}_{i,0} = \mathbf{N}$.

This branching program has length ℓ and width 2.

We can get a similar representation that handles a single conjunction of the literals (using de Morgan's law and the fact that Type II is closed under complement).

7 Application 1: Private constrained PRFs

In this section, we construct private constrained PRF for Type I branching programs (cf. Def. 6). The constructions are straight-forward adaptations of prior constructions in [CC17]; however, we require a different proof strategy which is reminiscent of that in [PS18].

7.1 Definitions

We recall the simulation-based definition for private constrained PRFs with one-key security from [CC17]. According to [CC17, Section 4.3], in the one-key setting the simulation-based definition is equivalent to the indistinguishability-based definition from [BLW17].

Definition 8 (Private constrained PRF (PCPRF))**.** Consider a family of functions $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda\in\mathbf{N}}$ where $\mathcal{F}_\lambda = \{F_k : D_\lambda \to R_\lambda\}$, along with a tuple of efficient functions (Gen, Constrain, Eval, Constrain.Eval). For a constraint family $\mathcal{C} = \{C_\lambda : D_\lambda \to \{0,1\}\}_{\lambda\in\mathbb{N}}$,

- The key generation algorithm $\mathsf{Gen}(1^\lambda, 1^\ell, \mathcal{F}_\lambda)$ takes the security parameter λ, the input length ℓ and the description of the constraint class $\mathcal{F}_\lambda$, generates the master secret key MSK.
- The evaluation algorithm $\mathsf{Eval}(\mathsf{MSK}, x)$ takes MSK, an input x, outputs $F_{\mathsf{MSK}}(x)$.
- The constraining algorithm $\mathsf{Constrain}(1^\lambda, \mathsf{MSK}, C)$ takes MSK, a constraint C, outputs the constrained key CK_C.
- The constrained evaluation algorithm $\mathsf{Constrain.Eval}(\mathsf{CK}_C, x)$ takes a constrained key CK_C, an input x, outputs $F_{\mathsf{CK}_C}(x)$.

$\mathcal{F}$ is a family of *private constrained PRF* for $\mathcal{C}$ if it satisfies the following properties:

Functionality preservation for $C(x) = 1$. For an input $x \in D_\lambda$ s.t. $C(x) = 1$,

$$\Pr[\mathsf{Eval}(\mathsf{MSK}, x) = \mathsf{Constrain.Eval}(\mathsf{CK}_C, x)] \geq 1 - \mathrm{negl}(\lambda),$$

where the probability is taken over the randomness in algorithms Gen and Constrain.

Pseudorandomness and constraint-hiding. For any polytime stateful algorithm Adv, there is a polytime stateful algorithm Sim such that:

$$\left\{\text{Experiment REAL}_{\mathsf{Adv}}(1^\lambda)\right\}_{\lambda\in\mathbb{N}} \approx_c \left\{\text{Experiment IDEAL}_{\mathsf{Adv},\mathsf{Sim}}(1^\lambda)\right\}_{\lambda\in\mathbb{N}}.$$

where the ideal and real experiments are defined as follows. In the experiments the adversary can ask a single constraint query followed by polynomially many input queries. Once Adv makes the constraint query $C \in \mathcal{C}_\lambda$, in the real experiment Adv obtains the constrained key generated by the constraining algorithm; in the ideal experiment Adv obtains a key generated by Sim, whereas Sim is given only the size of C. Once Adv makes an input query x, Adv is expected to provide a bit d_x indicating the value of $C(x)$. In the real experiment Adv obtains the unconstrained function value at x. In the ideal experiment Sim learns the indicator bit d_x; if $d_x = 1$ then Adv gets a value generated by Sim, and if $d_x = 0$ then Adv obtains a random value from the range R of the function. The output of the experiment is the final output bit of Adv.

Experiment $\text{REAL}_{\mathsf{Adv}}(1^\lambda)$	Experiment $\text{IDEAL}_{\mathsf{Adv},\mathsf{Sim}}(1^\lambda)$
$\mathsf{MSK} \leftarrow \mathsf{Gen}(1^\lambda)$,	$\mathsf{Sim} \leftarrow 1^\lambda$
$\mathsf{Adv} \rightarrow C$;	$\mathsf{Adv} \rightarrow C$;
$Adv \leftarrow \mathsf{Constrain}(\mathsf{MSK}, C)$	$\mathsf{Adv} \leftarrow \mathsf{Sim}(1^{\lvert C\rvert})$
Repeat :	*Repeat* :
$\mathsf{Adv} \rightarrow x;\ y = \mathsf{Eval}(\mathsf{MSK}, x)$	$\mathsf{Adv} \rightarrow x;\ y = \mathsf{Sim}(x, d_x)$
$\mathsf{Adv} \leftarrow y$	if $d_x = 0$ then $y = U(R)$; $\mathsf{Adv} \leftarrow y$
$\mathsf{Adv} \rightarrow b$; Output b	$\mathsf{Adv} \rightarrow b$; Output b

7.2 Construction

We construct private constrained PRFs for constraints recognizable by Type I branching programs.

For the ease of presentation we make notation conventions that are only applicable in Section 7.

For $n, t, w \in \mathbb{N}$ s.t. $t = wn + n$, for a matrix $\mathbf{X} \in \mathbb{Z}^{t\times *}$, let $\mathbf{X} = \begin{pmatrix}\overline{\mathbf{X}}\\ \underline{\mathbf{X}}\end{pmatrix}$ s.t. $\overline{\mathbf{X}} \in \mathbb{Z}^{(wn)\times *}$, $\underline{\mathbf{X}} \in \mathbb{Z}^{n\times *}$.

Construction 7.1 (PCPRF)**.** For a class of matrix branching programs $\mathcal{G}_{\mathbf{v},\varpi,w}$ that satisfies Definition 6, construct a family of private constraint PRFs for $\mathcal{G}_{\mathbf{v},\varpi,w}$ as follows.

Key Gen. $\mathsf{Gen}(1^\lambda, 1^\ell, \mathcal{G}_{\mathbf{v},\varpi,w})$ parses parameters $h, w \in \mathbb{N}$, $\mathbf{v} \in \{0,1\}^{1\times w}$ from $\mathcal{G}_{\mathbf{v},\varpi,w}$.

Sample parameters $n, m, p, q, B \in \mathbb{N}$, $\sigma \in \mathbb{R}^+$ according to Remark 7.2.

Sample $\{\mathbf{S}_{i,b} \leftarrow \chi^{n\times n}\}_{i\in[h],b\in\{0,1\}}$, $\mathbf{A}_h \xleftarrow{\$} \mathbb{Z}_q^{t\times m}$. Let $\mathsf{MSK} := \{\mathbf{S}_{i,b}\}_{i\in[h],b\in\{0,1\}}, \mathbf{A}_h$.

Eval. $\mathsf{Eval}(\mathsf{MSK}, \mathbf{x})$ parses MSK as $\{\mathbf{S}_{i,b}\}_{i\in[h],b\in\{0,1\}}, \mathbf{A}_h$, on input $\mathbf{x} \in \{0,1\}^\ell$, outputs

$$\left\lfloor \mathbf{S}_{\varpi(\mathbf{x})} \cdot \underline{\mathbf{A}}_h \right\rceil_p .$$

Constrain. $\mathsf{Constrain}(1^\lambda, \mathsf{MSK}, \Gamma)$ parses MSK as $\{\mathbf{S}_{i,b}\}_{i\in[h],b\in\{0,1\}}, \mathbf{A}_h$, the matrices in Γ as $\{\mathbf{M}_{i,b}\}_{i\in[h],b\in\{0,1\}}$, computes

$$(\mathbf{A}_0, \{\mathbf{D}_{i,b}\}_{i\in[h],b\in\{0,1\}}) \leftarrow \mathsf{ggh.encode}(\gamma_{\otimes\mathrm{diag}}, \{\mathbf{M}_{i,b}\}_{i\in[h],b\in\{0,1\}}, \{\mathbf{S}_{i,b}\}_{i\in[h],b\in\{0,1\}}, \mathbf{A}_h, \sigma)$$

and outputs

$$\mathsf{CK}_\Gamma = \mathbf{J} \cdot \mathbf{A}_0, \{\mathbf{D}_{i,b}\}_{i\in[h],b\in\{0,1\}} .$$

where $\mathbf{J} = (\mathbf{v} \otimes \mathbf{I}^{n\times n} \mid \mathbf{I}^{n\times n}) \in \{0,1\}^{n\times t}$.

Constrain Eval. $\mathsf{Constrain.Eval}(\mathsf{CK}_\Gamma, \mathbf{x})$ parses $\mathsf{CK}_\Gamma = \mathbf{J}\cdot\mathbf{A}_0, \{\mathbf{D}_{i,b}\}_{i\in[h],b\in\{0,1\}}$, on input $\mathbf{x} \in \{0,1\}^\ell$, outputs

$$\left\lfloor \mathbf{J} \cdot \mathbf{A}_0 \cdot \mathbf{D}_{\varpi(\mathbf{x})} \right\rceil_p .$$

Remark 7.2 (Parameters)**.** The parameters $n, m, p, q, B \in \mathbb{N}$, $\sigma \in \mathbb{R}^+$ are sampled under the following constraints for correctness and security. $m = \Omega(t \log q)$, $\sigma = \Omega(\sqrt{t \log q})$ for trapdoor functionality due to Lemma 3.10 and Lemma 3.11. $n = \Omega(\lambda \log q)$, $\chi = D_{\mathbb{Z},2\sqrt{\lambda}}$ for security due to Lemma 3.9 and Lemma 3.8. The noise threshold B is set according to Lemma 5.3

$$B \geq (w+1)\cdot h \cdot \left(m\sigma^2 \cdot \sqrt{n(w+1)}\right)^h \geq (w+1)\cdot h \cdot \left((w+1)\lambda \log^2 q\right)^{2.5h} .$$

Let $\epsilon \in (0,1)$. q and p are chosen s.t. $q \geq p \cdot B \cdot \omega(\mathrm{poly}(\lambda))$ for correctness, $q \leq (\sigma/\lambda)\cdot 2^{\lambda^{1-\epsilon}}$ for security due to Lemma 3.8.

Theorem 7.3 (Functionality and correctness)**.** *For all $x \in \{0,1\}^\ell$, with all but negligible probability*

$$\|\mathbf{J}\cdot\mathbf{A}_0\cdot\mathbf{D}_{\varpi(x)} - \left(\mathbf{S}_{\varpi(x)}\cdot\underline{\mathbf{A}}_h + (\mathbf{v}\otimes\mathbf{I})\cdot(\mathbf{M}_{\varpi(x)}\otimes\mathbf{S}_{\varpi(x)})\cdot\overline{\mathbf{A}}_h\right)\|_\infty \leq B \tag{11}$$

For $x \in \{0,1\}^\ell$ such that $C(x) = 1$, with all but negligible probability

$$\left\lfloor \mathbf{J}\cdot\mathbf{A}_0\cdot\mathbf{D}_{\varpi(\mathbf{x})}\right\rceil_p = \left\lfloor \mathbf{S}_{\varpi(\mathbf{x})}\cdot\underline{\mathbf{A}}_h\right\rceil_p .$$

Proof. The first statement follows Lemma 5.3. The second statement follows the setting of parameter $q = p\cdot B\cdot\omega(\mathrm{poly}(\lambda))$, and Def. 6 which says when $C(x)=1$, we have $\mathbf{vM}_{\varpi(x)} = 0^{1\times w}$. □

Example 7.1. For CNFs with ℓ variables, w clauses, encode them using Construction 6.4. Let $\mathbf{v} = \mathbf{1}^{1\times w}$, the resulting branching program satisfies the criteria for constructing PCPRFs.

Example 7.2. The "puncturing" functionality $f_{\mathbf{x}^*}(\mathbf{x}) = (\mathbf{x} \neq \mathbf{x}^*)$ can be instantiated with CNF with a single clause of disjunctions. To puncture on a point $\mathbf{x}^* \in \{0,1\}^\ell$, let $\Psi = (\bar{x}_1 \vee \bar{x}_2 \vee \ldots \vee \bar{x}_\ell)$. Concretely, we can take $w = 1$ and let $v = 1, \mathbf{M}_{i,x_i^*} = 1, \mathbf{M}_{i,1-x_i^*} = 0, \mathbf{P}_1 = 0, \mathbf{P}_0 = 1$, so that

$$\mathbf{M}_{\mathbf{x}} = \begin{cases} 0 & \text{if } \mathbf{x} \neq \mathbf{x}^* \\ 1 & \text{if } \mathbf{x} = \mathbf{x}^* \end{cases} .$$

7.3 Security proof

Next we prove Construction 7.1 satisfies constraint-hiding and pseudorandomness properties (cf. Definition 8).

To assist the proof we adapt theorems from the PRF construction of Boneh et al. [BLMR13, Theorems 4.3, 5.1]. For our purpose we need an unrounded version of their result. The proof is implicit in that of [BPR12, Theorem 5.2].

Lemma 7.4 (Implicit from [BPR12, BLMR13]). *Let $h, n, q, B \in \mathbb{N}$, $\sigma, \sigma^* \in \mathbb{R}$ s.t. $n = \Omega(\lambda \cdot \log q)$, $\sigma = \Omega(\sqrt{\lambda \cdot \log q})$, $\chi = D_{\mathbb{Z},\sigma}$. $B \geq h \cdot (\sqrt{n}\sigma)^h$, $\sigma^* > \omega(\mathrm{poly}(\lambda)) \cdot B$, $q \geq \sigma^* \cdot \omega(\mathrm{poly}(\lambda))$.*

Define a function family $\mathcal{F} = \left\{ f_{k(\lambda)} : \{0,1\}^{h(\lambda)} \to \mathbb{Z}_{q(\lambda)}^{n(\lambda)} \right\}_{\lambda \in \mathbb{N}}$, for which the key generation algorithm samples $\mathbf{a} \leftarrow U(\mathbb{Z}_q^n)$ as the private key, $\left\{ \mathbf{S}_{i,b} \leftarrow D_{\mathbb{Z},\sigma}^{n \times n} \right\}_{b \in \{0,1\}, i \in [h]}$ as the public parameters. The evaluation algorithm takes input $x \in \{0,1\}^h$, computes

$$f_{\mathbf{a}}(x) = \left(\prod_{i=1}^{h} \mathbf{S}_{i,x_i} \right) \cdot \mathbf{a} + \mathbf{E}_x = \mathbf{S}_x \cdot \mathbf{a} + \mathbf{E}_x \pmod q,$$

where $\mathbf{E}_x \leftarrow D_{\mathbb{Z},\sigma^*}^n$ is sampled freshly for every x.

Then assuming the hardness of $\mathsf{LWE}_{n,\mathrm{poly},q,U(\mathbb{Z}_q),D_{\mathbb{Z},\sigma},D_{\mathbb{Z},\sigma}}$. For $d = \mathrm{poly}(\lambda)$ different input queries $x^{[1]}$, ..., $x^{[d]}$, the outputs $f_{\mathbf{a}}(x^{[1]})$, ..., $f_{\mathbf{a}}(x^{[d]})$ are computationally indistinguishable from d independent uniformly random vectors from $\mathbb{Z}_q^n$.

Proof sketch. We first consider an expression analogous to the one in the proof of [BPR12, Lemma 5.5]

$$\begin{aligned} \tilde{f}_{\mathbf{a}}(x) :=\ & \left(\mathbf{S}_{1,x_1} \cdot \ldots \left(\mathbf{S}_{h-1,x_{h-1}} \cdot \left(\mathbf{S}_{h,x_h} \cdot \mathbf{a} + \mathbf{E}_{h,x_h} \right) + \mathbf{E}_{h-1,x_{h-1}} \right) \ldots + \mathbf{E}_{1,x_1} \right) + \mathbf{E}_x \\ =\ & f_{\mathbf{a}}(x) + \underbrace{\sum_{i=1}^{h} \left(\left(\prod_{j=1}^{i-1} \mathbf{S}_{j,x_j} \right) \cdot \mathbf{E}_{i,x_i} \right)}_{=:\mathbf{E}_x^*} \pmod q. \end{aligned} \tag{12}$$

where $\mathbf{E}_{1,x_1}, \ldots, \mathbf{E}_{h,x_h}$ are sampled independently from χ^n. Therefore we have $\|\mathbf{E}_x^*\|_\infty \leq B$, hence $\tilde{f}_{\mathbf{a}}(x) \approx_s f_{\mathbf{a}}(x)$ due to Lemma 3.2 and the setting of parameters $\sigma^* > \omega(\mathrm{poly}(\lambda)) \cdot B$, $\mathbf{E}_x \leftarrow D_{\mathbb{Z},\sigma^*}^n$ (in the proof of [BPR12, Theorem 5.2] the additional terms are added due to rounding.)

Once we have Eqn. (12) the rest of the proof follows the proof of [BPR12, Theorem 5.2], except that we use $\mathsf{LWE}_{n,\mathrm{poly},q,U(\mathbb{Z}_q),D_{\mathbb{Z},\sigma},D_{\mathbb{Z},\sigma}}$ instead of $\mathsf{LWE}_{n,\mathrm{poly},q,D_{\mathbb{Z},\sigma},U(\mathbb{Z}_q),D_{\mathbb{Z},\sigma}}$. □

Theorem 7.5. *Construction 7.1 is a private-constrained PRF assuming $\mathsf{LWE}_{n,\mathrm{poly},q,U(\mathbb{Z}_q),D_{\mathbb{Z},\sigma},D_{\mathbb{Z},\sigma}}$.*

Proof. The simulator $\mathsf{Sim}(1^\lambda)$ proceeds as follows:

1. Preprocessing: Sample $\mathbf{U} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$, $\left\{ \mathbf{D}_{i,b} \leftarrow D_{\mathbb{Z},\sigma}^{m \times m} \right\}_{i \in [h], b \in \{0,1\}}$.
2. Given the constrained key query, Sim outputs $\mathbf{U}, \{\mathbf{D}_{i,b}\}_{i \in [h], b \in \{0,1\}}$ as the simulated constrained key.

3. Given an input query x with indicator d_x, Sim outputs

$$F_{\mathsf{Sim}}(x) = \begin{cases} \lfloor \mathbf{U} \cdot \mathbf{D}_{\varpi(x)} \rceil_p & \text{if } d_x = 1 \\ \mathbf{Y} \xleftarrow{\$} \mathbb{Z}_p^{n \times m} & \text{if } d_x = 0 \end{cases}.$$

To prove the real experiment is indistinguishable from the simulated one, we introduce an intermediate simulator Sim^*, which differs from Sim in the response to the evaluation queries. $\mathsf{Sim}^*(1^\lambda)$ proceeds as follows:

1. Preprocessing: Sample $\mathbf{U} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$, $\left\{\mathbf{D}_{i,b} \leftarrow D_{\mathbb{Z},\sigma}^{m \times m}\right\}_{i \in [h], b \in \{0,1\}}$.
2. Given the constrained key query, Sim^* outputs $\mathbf{U}, \{\mathbf{D}_{i,b}\}_{i \in [h], b \in \{0,1\}}$ as the simulated constrained key.
3. Given an input query x with indicator d_x, Sim^* outputs

$$F_{\mathsf{Sim}^*}(x) = \begin{cases} \lfloor \mathbf{U} \cdot \mathbf{D}_{\varpi(x)} \rceil_p & \text{if } d_x = 1 \\ \lfloor \mathbf{U} \cdot \mathbf{D}_{\varpi(x)} - (\mathbf{v} \otimes \mathbf{I}) \cdot (\mathbf{M}_{\varpi(x)} \otimes \mathbf{S}_{\varpi(x)}) \cdot \overline{\mathbf{A}}_h \rceil_p & \text{if } d_x = 0 \end{cases}.$$

Lemma 7.6. *The real distribution is indistinguishable from the output of* Sim^* *assuming* $\mathsf{LWE}_{n,2m,q,U(\mathbb{Z}_q),D_{\mathbb{Z},\sigma},D_{\mathbb{Z},\sigma}}$.

Proof. We prove indistinguishability assuming semantic security of the $\gamma_{\otimes\text{diag}}$-GGH encodings with auxiliary input $\{\mathbf{S}_{i,b}\}_{i \in [h], b \in \{0,1\}}, \mathbf{J} \cdot \mathbf{A}_0, \overline{\mathbf{A}}_h$ as shown in Lemma 5.9. That is, we show if there exists a distinguisher A for the output distributions of the real world versus Sim^*, we build a distinguisher A' for the experiments in Lemma 5.9.

1. Once A makes a constraint key query with branching program $\Gamma_{\mathbf{v},\varpi,w}$, A' calls for a sample from the experiments in Lemma 5.9 with the matrices $\{\mathbf{M}_{i,b}\}_{i \in [h], b \in \{0,1\}}$ and vector $\mathbf{v}$ from $\Gamma_{\mathbf{v},\varpi,w}$, get back auxiliary inputs $\{\mathbf{S}_{i,b}\}_{i \in [h], b \in \{0,1\}}, \mathbf{J} \cdot \mathbf{A}_0, \overline{\mathbf{A}}_h$, where $\mathbf{J} = (\mathbf{v} \mid 1) \otimes \mathbf{I}^{n \times n}$ and $\{\mathbf{D}_{i,b}\}_{i \in [h], b \in \{0,1\}}$ sampled either from the real (GGH15 encoding) or the Gaussian distribution. A' then send $\mathbf{J} \cdot \mathbf{A}_0$, $\{\mathbf{D}_{i,b}\}_{i \in [h], b \in \{0,1\}}$ as the response for the constrained key.
2. Once A makes an evaluation query x with indicator d_x, A' responses with

$$\begin{cases} \lfloor \mathbf{J} \cdot \mathbf{A}_0 \cdot \mathbf{D}_{\varpi(x)} \rceil_p & \text{if } d_x = 1 \\ \lfloor \mathbf{J} \cdot \mathbf{A}_0 \cdot \mathbf{D}_{\varpi(x)} - (\mathbf{v} \otimes \mathbf{I}) \cdot (\mathbf{M}_{\varpi(x)} \otimes \mathbf{S}_{\varpi(x)}) \cdot \overline{\mathbf{A}}_h \rceil_p & \text{if } d_x = 0 \end{cases}.$$

   This is where we use the auxiliary input $\{\mathbf{S}_{i,b}\}_{i \in [h], b \in \{0,1\}}, \mathbf{J} \cdot \mathbf{A}_0, \overline{\mathbf{A}}_h$.

First we observe that $\mathbf{J} \cdot \mathbf{A}_0$ distributes identically to $\mathbf{U}$ from the output of Sim^*. Then A' responses of evaluation queries is exactly the same to the responses of Sim^*, and same to the responses in the real world up to negligible statistical error due to Theorem 7.3. In addition, if $\{\mathbf{D}_{i,b}\}_{i \in [h], b \in \{0,1\}}$ are sampled the real distribution (GGH15 encoding), it corresponds to the distribution in the real constrained key; if $\{\mathbf{D}_{i,b}\}_{i \in [h], b \in \{0,1\}}$ are sampled the Gaussian distribution, it corresponds to the output of Sim^*. We conclude that the advantage of A' is the same of the advantage of A. □

Lemma 7.7. *The output of* Sim^* *is indistinguishable from the output of* Sim *assuming* $\mathsf{LWE}_{n,\mathrm{poly},q,U(\mathbb{Z}_q),D_{\mathbb{Z},\sigma},D_{\mathbb{Z},\sigma}}$.

Proof. The constrained keys simulated by Sim^* and Sim are the same. It remains to prove the PRF evaluations produced by Sim^* on input x such that $C(x) = 0$ are indistinguishable from uniformly random. To assist the proof we pick $\sigma^* \in \mathbb{R}^+$ s.t. $h \cdot (\sqrt{n}\sigma)^h \cdot \omega(\mathrm{poly}(\lambda)) < \sigma^* < \frac{q}{p \cdot \omega(\mathrm{poly}(\lambda))}$.

For $x \in \{0,1\}^\ell$ such that $C(x) = 0$, rearrange the output produced by Sim^*:

$$\begin{aligned} F_{\mathsf{Sim}^*}(x) = & \left\lfloor \mathbf{U} \cdot \mathbf{D}_{\varpi(x)} - (\mathbf{v} \otimes \mathbf{I}) \cdot (\mathbf{M}_{\varpi(x)} \otimes \mathbf{S}_{\varpi(x)}) \cdot \overline{\mathbf{A}}_h \right\rceil_p \\ = & \left\lfloor \mathbf{U} \cdot \mathbf{D}_{\varpi(x)} - \left((\mathbf{v} \cdot \mathbf{M}_{\varpi(x)}) \otimes \mathbf{S}_{\varpi(x)}\right) \cdot \overline{\mathbf{A}}_h \right\rceil_p \\ = & \left\lfloor \mathbf{U} \cdot \mathbf{D}_{\varpi(x)} - \sum_{j=1}^{w} \left((\mathbf{v} \cdot \mathbf{M}_{\varpi(x)})_j \cdot \mathbf{S}_{\varpi(x)} \cdot \overline{\mathbf{A}}_h^{(j)} \right) \right\rceil_p \\ \approx_s & \left\lfloor \mathbf{U} \cdot \mathbf{D}_{\varpi(x)} - \sum_{j=1}^{w} \left((\mathbf{v} \cdot \mathbf{M}_{\varpi(x)})_j \cdot \mathbf{S}_{\varpi(x)} \cdot \overline{\mathbf{A}}_h^{(j)} \right) + \mathbf{E}_x \right\rceil_p \end{aligned} \tag{13}$$

where $(\mathbf{v} \cdot \mathbf{M}_{\varpi(x)})_j$ denotes the j^{th} coordinate of $\mathbf{v} \cdot \mathbf{M}_{\varpi(x)}$; $\overline{\mathbf{A}}_h^{(j)}$ denotes the $((j-1)n+1)^{th}$ to $(jn)^{th}$ rows of $\overline{\mathbf{A}}_h$; $\mathbf{E}_x \leftarrow D_{\mathbb{Z},\sigma^*}^{n \times m}$. The setting of σ^* guarantees the statistical closeness of the last two distributions.

The functionality of the Type I branching program Γ guarantees that $\mathbf{v} \cdot \mathbf{M}_{\varpi(x)} \in \{0,1\}^{1 \times w} \setminus \{\mathbf{0}^{1 \times w}\}$, which means at least 1 term in the summation is non-zero. Suppose the non-zero term is j^*, we will use $\overline{\mathbf{A}}_h^{(j^*)}$ as the secret to trigger Lemma 7.4.

Formally, let $\mathcal{X}_0 := \{x^{[1]}, ..., x^{[d]}\}$ be the set of $d = \mathrm{poly}(\lambda)$ input queries such that $C(x^{[i]}) = 0$, $\forall i \in [d]$. We run through w hybrid distributions. In the j^{th} hybrid, $j \in [w]$, we take out the inputs $\mathcal{X}_j^* := \{x^* \mid (\mathbf{v} \cdot \mathbf{M}_{\varpi(x^*)})_j = 1, x^* \in \mathcal{X}_{j-1}\}$, set $\mathcal{X}_j := \mathcal{X}_{j-1} \setminus \mathcal{X}_j^*$. For all $x^* \in \mathcal{X}_j^*$, we have

$$F_{\mathsf{Sim}^*}(x^*) \approx_s \left\lfloor \mathbf{Z}_{x^*} - (\mathbf{S}_{\varpi(x^*)} \cdot \overline{\mathbf{A}}_h^{(j)} - \mathbf{E}_{x^*}) \right\rceil_p \approx_c \lfloor \mathbf{Z}_{x^*} - \mathbf{U}_{x^*} \rceil_p \tag{14}$$

where $\mathbf{Z}_{x^*}$ denotes the rest of the terms from the last expression of Eqn. (13), $\mathbf{U}_{x^*} \leftarrow U(\mathbb{Z}_q^{n \times m})$, independently for each x^*. The last $\approx_c$ follows Lemma 7.4 by treating $\overline{\mathbf{A}}_h^{(j)}$ as the secret.

At the end of hybrid w we finish with an empty set $\mathcal{X}_w$. The proof of Lemma 7.7 concludes. □

The proof completes by combining the Lemmas 7.6 and 7.7. □

8 Application 2: Lockable obfuscation

In this section, we construct lockable obfuscation for general Type II branching programs (cf. Def 7)

Lockable (a.k.a. compute-and-compare) obfuscation [GKW17a, WZ17] takes as inputs a program f, a target bit-string $\mathbf{y}$, a message μ, produce an obfuscated program $\mathsf{Obf}[f, \mathbf{y}, \mu]$ that performs the following "compute-and-compare" functionality: on input x, if $f(x) = \mathbf{y}$, outputs μ; otherwise outputs $\perp$. The security requirement is that of distributional virtual-black-box (VBB) security, which guarantees that the obfuscated program does not reveal any partial information about $f, \mathbf{y}, \mu$ as long as they are chosen from some distribution where $\mathbf{y}$ has sufficient pseudo-entropy given f.

We describe a basic construction of lockable obfuscation, where "basic" means we work with a fixed message $\mu = 1$, and VBB security holds when $\mathbf{y}$ is sampled uniformly random. The point however is that we can handle a family of functions $\mathcal{F} = \{f\}$ represented by more general branching programs, as opposed to the basic constructions in [GKW17a, WZ17] where $\mathcal{F}$ must be recognized by permutation matrix branching programs. The extensions to multi-bit message μ, target $\mathbf{y}$ with reasonable pseudo-entropy, program f represented by general circuits or Turing machines follow the analysis and transformations in [GKW17a, WZ17].

Both the construction and the proof are straight-forward adaptations of the prior constructions in [GKW17a, WZ17], once we have established semantic security of the underlying generalized GGH15 encodings as in Lemma 5.9.

8.1 Definition

Recall the definition from [GKW17a, WZ17].

Definition 9 (Lockable (or compute-and-compare) obfuscation)**.** Consider a family of functions $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{N}}$ where $\mathcal{F}_\lambda = \left\{f : \{0,1\}^{\ell(\lambda)} \to \{0,1\}^{\nu(\lambda)}\right\}$, $\nu(\lambda) = \omega(\log\lambda)$. A lockable obfuscator takes a function $f \in \mathcal{F}$ and a target $\mathbf{y} \in \{0,1\}^\nu$, outputs an obfuscated program $\mathsf{Obf}[f,\mathbf{y}]$ which satisfies the following properties:

Functionality. $\mathsf{Obf}[f,\mathbf{y}]$ takes an input $x \in \{0,1\}^\ell$, output 1 if $f(x) = \mathbf{y}$; $\perp$ elsewhere.

Virtual black-box security. A lockable obfuscator is said to satisfy virtual black-box security if there is a p.p.t. simulator S such that for all $f \in \mathcal{F}$,

$$\mathsf{Obf}[f,\mathbf{y}] \approx_c S(1^\lambda, 1^{|f|})$$

over $\mathbf{y} \xleftarrow{\$} \{0,1\}^\nu$ and the randomness of the obfuscator and S.

8.2 Construction

We construct lockable obfuscation for function families $\mathcal{F}_\lambda = \left\{f_\lambda : \{0,1\}^{\ell(\lambda)} \to \{0,1\}^{\nu(\lambda)}\right\}$, where $\nu(\lambda) \geq 2n\log q$, the functionality of each output bit can be recognized by a Type II branching program (cf. Def. 7).

For the ease of presentation we make notation conventions that are only applicable in Section 8. For $n,t,w,\nu \in \mathbb{N}$ such that $t = \nu wn + n$. For a matrix $\mathbf{X} \in \mathbb{Z}^{t\times *}$, let $\mathbf{X} = \begin{pmatrix}\overline{\mathbf{X}}\\ \underline{\mathbf{X}}\end{pmatrix}$ s.t. $\overline{\mathbf{X}} \in \mathbb{Z}^{(\nu wn)\times *}$, $\underline{\mathbf{X}} \in \mathbb{Z}^{n\times *}$. For $i \in [\nu]$, $j \in [w]$, let $\overline{\mathbf{X}}^{(i)}$ denote the $((i-1)wn+1)^{th}$ to $(iwn)^{th}$ rows of $\overline{\mathbf{X}}$; $\overline{\mathbf{X}}^{(i,j)}$ denote the $((j-1)n+1)^{th}$ to $(jn)^{th}$ rows of $\overline{\mathbf{X}}^{(i)}$. For arbitrary variable (say Z) not representing a matrix (or a column vector) with νwn rows, we use the superscript $Z^{(i)}$ to denote objects related to the i^{th} output bit of the function $f \in \mathcal{F}$. For example, the functionality of the i^{th} output bit of f is denoted as $f^{(i)}$.

Construction 8.1 (Lockable obfuscation)**.** For a function family $\mathcal{F} = \left\{f : \{0,1\}^\ell \to \{0,1\}^\nu\right\}$ s.t. for all $f \in \mathcal{F}$, $f^{(i)}$ can be recognized by a branching program Γ from the class $\mathcal{G}_{\mathbf{v},\varpi,w}$ that satisfies Definition 7. Construct the lockable obfuscation for $\mathcal{F}$ as follows.

Parameter Gen. $\mathsf{Gen}(1^\lambda, 1^\ell, 1^\nu, \mathcal{G}_{\mathbf{v},\varpi,w})$ parses parameters $h, w \in \mathbb{N}$, $\mathbf{v} \in \{0,1\}^{1\times w}$ from $\mathcal{G}_{\mathbf{v},\varpi,w}$. Sample parameters $n, m, q, B \in \mathbb{N}$, $\sigma \in \mathbb{R}^+$ according to Remark 8.2.

Sample $\{\mathbf{S}_{i,b} \leftarrow \chi^{n\times n}\}_{i\in[h],b\in\{0,1\}}$. Sample a column vector $\overline{\mathbf{A}}_h \xleftarrow{\$} \mathbb{Z}_q^{\nu wn}$.

Obfuscate. Given $\mathbf{y} \leftarrow \{0,1\}^\nu$, sample $\underline{\mathbf{A}}_h := \sum_{i\in[\nu]} \overline{\mathbf{A}}_h^{(i,(2-y_i))}$.

Given $f \in \mathcal{F}$ where for $i \in [\nu]$, each $f^{(i)}$ is recognized by $\Gamma^{(i)} = \left\{\varpi, \left\{\mathbf{M}_{k,b}^{(i)}\right\}_{k\in[h],b\in\{0,1\}}, \mathcal{P}_0^{(i)}, \mathcal{P}_1^{(i)}\right\}$.

For $k \in [h], b \in \{0,1\}$. Define $\mathbf{M}_{k,b} \in \{0,1\}^{(\nu w)\times(\nu w)}$ by assigning $\mathbf{M}_{k,b}^{(i)}$ in the i^{th} diagonal w-block, i.e.

$$\mathbf{M}_{k,b} = \mathrm{diag}(\mathbf{M}_{k,b}^{(1)}, ..., \mathbf{M}_{k,b}^{(\nu)}).$$

Now compute (following Construction 5.1 under Remark 5.4)

$$(\mathbf{A}_0, \{\mathbf{D}_{i,b}\}_{i\in[h],b\in\{0,1\}}) \leftarrow \mathsf{ggh.encode}(\gamma_{\otimes\mathrm{diag}}, \{\mathbf{M}_{i,b}\}_{i\in[h],b\in\{0,1\}}, \{\mathbf{S}_{i,b}\}_{i\in[h],b\in\{0,1\}}, \mathbf{A}_h, \sigma)$$

and output $\mathsf{Obf}[f,\mathbf{y}] := \mathbf{J}\mathbf{A}_0, \{\mathbf{D}_{i,b}\}_{i\in[h],b\in\{0,1\}}$ where $\mathbf{J} := (\mathbf{v} \mid ... \mid \mathbf{v} \mid -1)\otimes\mathbf{I}^{n\times n} \in \{0,1\}^{n\times t}$.

Eval. $\mathsf{Eval}(\mathsf{Obf}[f,\mathbf{y}], x)$ parses the obfuscated code as $\mathbf{J}\mathbf{A}_0, \{\mathbf{D}_{i,b}\}_{i\in[h],b\in\{0,1\}}$, outputs

$$\mathsf{Eval}(\mathsf{Obf}[f,\mathbf{y}], x) = \begin{cases} 1 & \text{if } \|\mathbf{J}\cdot\mathbf{A}_0\cdot\mathbf{D}_{\varpi(x)}\|_\infty \le B \\ \perp & \text{else} \end{cases}.$$

Remark 8.2 (Parameters)**.** The parameters $n, m, q, B \in \mathbb{N}$, $\sigma \in \mathbb{R}^+$ are sampled under the following constraints for correctness and security. $m = \Omega(t \log q)$, $\sigma = \Omega(\sqrt{t \log q})$ for trapdoor functionality due to Lemma 3.10 and Lemma 3.11. $n = \Omega(\lambda \log q)$, $\chi = D_{\mathbb{Z},2\sqrt{\lambda}}$ for security due to Lemma 3.9 and Lemma 3.8. The noise threshold B is set according to Lemma 5.3

$$B \ge (\nu w+1)\cdot h\cdot\left(m\sigma^2\cdot\sqrt{n(\nu w+1)}\right)^h \ge (\nu w+1)\cdot h\cdot\left((\nu w+1)\lambda\log^2 q\right)^{2.5h}.$$

Let $\epsilon \in (0,1)$. q is chosen s.t. $q \ge B\cdot\omega(\mathrm{poly}(\lambda))$ for correctness, $q \le (\sigma/\lambda)\cdot 2^{\lambda^{1-\epsilon}}$ for security due to Lemma 3.8.

Remark 8.3. The construction can be generalized to handle different dimensions w, different vectors $\mathbf{v}$ for the branching program of each output bit.

Example 8.1 (A special compute-and-compare functionality)**.** Consider lockable obfuscation for the following family of functions $\mathcal{F} = \{f : \{0,1\}^\ell \to \{0,1\}^\nu\}$ where each output bit $f^{(i)}$ of f is either a conjunction or a disjunction of an arbitrary subset of the literals $x_1, \ldots, x_\ell$ and their negations $\bar{x}_1, \ldots, \bar{x}_\ell$. Then, each $f^{(i)}$ is computable by a Type II matrix branching program $\Gamma^{(i)}$ of length ℓ using Construction 6.5 with $w = 2, \mathbf{v} = (0 \mid 1)$. On the other hand, if we are restricted to permutation branching programs, then we would require length ℓ^2. Therefore, for this family of functions, we achieve more efficient lockable obfuscation than the prior construction [WZ17].

Remark 8.4 (Relation to obfuscating conjunctions in [BR13, BVWW16, WZ17])**.** Brakerski, et al. [BR13, BVWW16, WZ17] considered the problem of obfuscating conjunctions in the distributional virtual black-box setting. This problem can be reduced to lockable obfuscation for a sub-class of the function family described in Example 8.1 where each $f^{(i)}$ is either 1 or x_i. Observe that such $f^{(i)}$ can be computed by a permutation branching program of length ℓ, so using non-permutation matrices does not improve over that in [WZ17].

Theorem 8.5 (Correctness). *Construction 8.1 satisfies statistical correctness. Namely, for all f, $\mathbf{y} \in \{0,1\}^\nu$, $x \in \ell$, with all but negligible probability over the randomness of the obfuscator,*

$$\mathsf{Eval}(\mathsf{Obf}[f,\mathbf{y}],x) = \begin{cases} 1 & \text{if } f(x) = \mathbf{y} \\ \perp & \text{if } f(x) \neq \mathbf{y} \end{cases}.$$

Proof. For $x \in \{0,1\}^\ell$, $\mathbf{y} \in \{0,1\}^\nu$,

$$\begin{aligned} \mathbf{J} \cdot \mathbf{A}_0 \cdot \mathbf{D}_{\varpi(x)} &= \sum_{i=1}^{\nu} \left((\mathbf{v} \otimes \mathbf{I}^{n\times n}) \cdot (\mathbf{M}^{(i)}_{\varpi(x)} \otimes \mathbf{S}_{\varpi(x)}) \cdot \overline{\mathbf{A}}^{(i)}_h \right) - \mathbf{S}_{\varpi(x)} \cdot \underline{\mathbf{A}}_h + \mathbf{E}^* \\ &= \sum_{i=1}^{\nu} \left(\mathbf{S}_{\varpi(x)} \cdot \overline{\mathbf{A}}^{(i,j^{(i)})}_h \right) - \mathbf{S}_{\varpi(x)} \cdot \underline{\mathbf{A}}_h + \mathbf{E}^* \qquad (15) \\ &= \mathbf{S}_{\varpi(x)} \cdot \sum_{i=1}^{\nu} \left(\overline{\mathbf{A}}^{(i,j^{(i)})}_h - \overline{\mathbf{A}}^{(i,(2-y_i))}_h \right) + \mathbf{E}^* \end{aligned}$$

where $\mathbf{E}^* \in \mathbb{Z}^n$ s.t. $\|\mathbf{E}^*\|_\infty \leq B$; $j^{(i)} := \begin{cases} 1 & \text{if } \mathbf{v}\mathbf{M}^{(i)}_{\varpi(x)} = \mathbf{e}_1 \\ 2 & \text{if } \mathbf{v}\mathbf{M}^{(i)}_{\varpi(x)} = \mathbf{e}_2 \end{cases}$. In Eqn. (15), the first equality follows Lemma 5.3; the second one follows the structural criteria of Type II branching programs (cf. Def 7); the third one is due to the construction of $\underline{\mathbf{A}}_h$.

If $f(x) = \mathbf{y}$, then for all $i \in [\nu]$, we have $j^{(i)} = 2 - y_i$. Hence

$$\mathbf{J} \cdot \mathbf{A}_0 \cdot \mathbf{D}_{\varpi(x)} = \mathbf{S}_{\varpi(x)} \cdot \mathbf{0}^n + \mathbf{E}^* = \mathbf{E}^*$$

On the other hand, if $f(x) \neq \mathbf{y}$, then the set $\mathcal{I} = \left\{ i \mid i \in [\nu] \text{ s.t. } j^{(i)} \neq 2 - y_i \right\}$ is not empty. Continue with Eqn. (15)

$$\mathbf{J} \cdot \mathbf{A}_0 \cdot \mathbf{D}_{\varpi(x)} = \mathbf{S}_{\varpi(x)} \cdot \underbrace{\sum_{i\in\mathcal{I}} \left(\overline{\mathbf{A}}^{(i,j^{(i)})}_h - \overline{\mathbf{A}}^{(i,(2-y_i))}_h \right)}_{=:\mathbf{A}^{(\mathcal{I})}_h} + \mathbf{E}^* \qquad (16)$$

Since $\mathcal{I}$ is not empty, $\mathbf{A}^{(\mathcal{I})}_h$ is the sum of independent uniformly random vectors over $\mathbb{Z}^n_q$, so $\mathbf{A}^{(\mathcal{I})}_h$ distributes uniformly random over $\mathbb{Z}^n_q$. At the same time we have $\mathbf{S}_{\varpi(x)}$ is invertible (over $\mathbb{Z}_q$) with probability $1 - \mathrm{negl}(\lambda)$ due to [WZ17, Claims 2.7, 4.11]. So $\mathbf{S}_{\varpi(x)} \cdot \mathbf{A}^{(\mathcal{I})}_h$ distributes uniformly random over $\mathbb{Z}^n_q$. Hence $\Pr\left[\|\mathbf{J} \cdot \mathbf{A}_0 \cdot \mathbf{D}_{\varpi(x)}\|_\infty \leq B\right] \leq B/q = \mathrm{negl}(\lambda)$ where the probability is taken over the randomness of the obfuscator. □

8.3 Security proof

Theorem 8.6. *Construction 8.1 satisfies VBB security assuming* $\mathsf{LWE}_{n,2m,q,U(\mathbb{Z}_q),D_{\mathbb{Z},\sigma},D_{\mathbb{Z},\sigma}}$.

Proof. The simulator Sim simply samples $\mathbf{U} \xleftarrow{\$} \mathbb{Z}^{n\times m}_q$, $\left\{ \mathbf{D}_{i,b} \leftarrow D^{m\times m}_{\mathbb{Z},\sigma} \right\}_{i\in[h-1],b\in\{0,1\}}$, $\mathbf{D}_{h,0}, \mathbf{D}_{h,1} \leftarrow D^m_{\mathbb{Z},\sigma}$, and output them as the simulated obfuscation.

To prove that the simulated distribution is indistinguishable from the real, we introduce an intermediate simulator Sim* who gets f but not $\mathbf{y}$. Sim* runs the same obfuscation procedure as

the real world, except that Sim^* samples $\underline{\mathbf{A}}_h \xleftarrow{\$} \mathbb{Z}_q^n$, instead of $\underline{\mathbf{A}}_h := \sum_{i\in[\nu]} \overline{\mathbf{A}}_h^{(i,(2-y_i))}$ in the real world. Since $\mathbf{y}$ distributes uniformly random over $\{0,1\}^\nu$, $\nu \geq 2n\log q$. The output of Sim^* is statistically close to the real distribution due to leftover hash lemma, cf. Lemma 3.1.

Finally, the output of Sim^* and Sim are indistinguishable assuming $\mathsf{LWE}_{n,2m,q,U(\mathbb{Z}_q),D_{\mathbb{Z},\sigma},D_{\mathbb{Z},\sigma}}$ due to the semantic security of the $\gamma_{\otimes\text{diag}}$-GGH encodings with auxiliary input $\mathbf{J}\cdot\mathbf{A}_0$, cf. Lemma 5.9. □

9 New attacks to iO candidates for branching programs

We extend the applicability of cryptanalytic attacks on iO candidates. The new attack algorithm simply computes the rank of a matrix formed by honest evaluations of the obfuscated code, then uses the rank as the distinguishing factor. So we name the new attack as a "rank attack".

To classify the attacks we introduce additional terminologies for matrix branching programs.

Definition 10 (Single/multi-input)**.** A branching program is called *single-input* if each step of the program is controlled by 1 bit from the input. A branching program is called *dual-input* or even *multi-input* if each step of the branching program is controlled by 2 or more bits from the input.

The branching programs following Def. 5 are implicitly single-input due to the definition of input mappings ι, ϖ. By default we work with single-input BPs.

Definition 11 (Input partition)**.** A branching program Γ is (s, L)-*input-partitioned* if the BP steps can be partitioned into s consecutive intervals $[h] = \mathcal{H}_1 \mid \mathcal{H}_2 \mid \ldots \mid \mathcal{H}_s$. In each interval, there are L input bits that control only steps in that interval and nowhere else. When (s, L) are dropped, it indicates that $s \geq 2$, $L \geq 1$.

Definition 12 (Input repetition)**.** A branching program Γ is c-*input-repeating* if *all* of its input bits appear at least in c non-consecutive intervals of steps.

For example, a 2-input-repeating branching program is not input-partitioned. A read-once branching program is 1-input-repeating and (s, L)-input-partitioned as long as $s \cdot L \leq h$. An oblivious permutation branching program obtained from Barrington's theorem over a depth-d circuit with ℓ input bits is $O(4^d)$-input-repeating.

9.1 The description of the iO candidates

We describe the obfuscation framework of Garg et al. [GGH+13b] in a way that is easy to generalize to other variants in the literature. On a high level, the existing candidates first fix a plaintext matrix branching program; then add several randomization layers, also known as "safeguards"; finally encode the randomized matrices using the graded encoding schemes. Different candidates make different choices in these steps. As examples, here are some of the iO candidates for branching programs interpreted in this framework.

Construction	Matrix type	Diagonal padding	Scalar & Kilian	Mmaps
[GGH+13b]	Permutation	Yes	Yes	Not specified
[BGK+14]	Dual-input permutation	No	Yes	Not specified
[GMM+16]	Dual-input permutation	Yes	Yes	GGH13
[HHSS17]	Read-once, non-permutation	Yes	No	GGH15

From now we only discuss single-input BPs. Let R denote the base ring (the typical choices are $R = \mathbb{Z}$ and $R = \mathbb{Z}[X]/(\phi(x))$ for an irreducible polynomial ϕ). The obfuscator works as follows:

0. Convert the function into a matrix branching program. The obfuscator first converts f into a branching program $\Gamma = \left\{\varpi,\ \{\mathbf{M}_{i,b} \in \{0,1\}^{w\times w}\}_{i\in[h],b\in\{0,1\}}, \mathcal{P}_0, \mathcal{P}_1\right\}$. A typical choice of the target sets is

$$\mathcal{P}_0 = \left\{\mathbf{I}^{w\times w}\right\}, \quad \mathcal{P}_1 = \{0,1\}^{w\times w} \setminus \mathcal{P}_0. \tag{17}$$

1. Add safeguards (randomizations). The randomization step embeds every matrix $\mathbf{M}_{i,b}$ inside a higher-dimension randomized matrix $\hat{\mathbf{S}}_{i,b}$, in an attempt to prevent leaking information from illegal evaluations, at the same time randomize the plaintext matrices as much as possible. The functionality requirement is, for $x \in \{0,1\}^\ell$ we have whp

$$\begin{cases} \mathbf{J}\cdot\hat{\mathbf{S}}_{\varpi(x)}\cdot\mathbf{L} = 0 & \text{if } f(x) = 0 \\ \mathbf{J}\cdot\hat{\mathbf{S}}_{\varpi(x)}\cdot\mathbf{L} \neq 0 & \text{if } f(x) = 1 \end{cases}$$

where $\mathbf{J}$ is a row vector and $\mathbf{L}$ is a column vector. We will also refer to $\mathbf{J}, \mathbf{L}$ as the left and right bookends.

Here we describe a typical approach of producing $\hat{\mathbf{S}}_{i,b}$ that is used in [GGH+13b, GGH15].

Add a dummy program. Add a "dummy branch" Γ', which is a branching program with the same input mapping function ϖ consisting of only identity matrices of the same dimension as the ones in the functional branch. Namely $\mathbf{M}'_{i,b} = \mathbf{I}^{w\times w}$.

Diagonal paddings and bookends. Let $t = 2w + 2n$. For each $i \in [h], b \in \{0,1\}$, choose random matrices $\mathbf{R}_{i,b}, \mathbf{R}'_{i,b} \in R^{n\times n}$. Pick vectors $\mathbf{r}_J, \mathbf{r'}_J, \mathbf{r}_L, \mathbf{r'}_L$ randomly from $R^{1\times w}$ subject to the following condition:

$$\text{for all } x \in \{0,1\}^\ell \text{ s.t. } f(x) = 0,\ \ \mathbf{r}_J\mathbf{M}_{\varpi(x)}\mathbf{r}_L^T - \mathbf{r'}_J\mathbf{M}'_{\varpi(x)}\mathbf{r'}_L^T = 0.$$

Then create "bookends" $\mathbf{J} \in R^{1\times t}, \mathbf{L} \in R^{t\times 1}$ such that

$$\mathbf{J} = (\mathbf{r}_J \mid \mathbf{0}^{1\times n} \mid \mathbf{r'}_J \mid \mathbf{0}^{1\times n}), \text{ and } \mathbf{L} = (\mathbf{r}_L \mid \mathbf{1}^{1\times n} \mid -\mathbf{r'}_L \mid \mathbf{1}^{1\times n})^T.$$

Kilian-style randomization and bundling scalars. Sample bundling scalars $\left\{\alpha_{i,b}, \alpha'_{i,b}\right\}_{i\in[h],b\in\{0,1\}}$ and invertible matrices $\left\{\mathbf{K}_i, \mathbf{K'}_i \in R^{(w+n)\times(w+n)}\right\}_{i\in[h-1]}$, identity matrices $\mathbf{K}_0, \mathbf{K}_h, \mathbf{K'}_0, \mathbf{K'}_h$. The scalars are chosen under the constraint that for any input bit $j \in [\ell]$,

$$\prod_{\iota(i)=j} \alpha_{i,0} = \prod_{\iota(i)=j} \alpha'_{i,0} \text{ and } \prod_{\iota(i)=j} \alpha_{i,1} = \prod_{\iota(i)=j} \alpha'_{i,1}. \tag{18}$$

We sometime use the notations

$$\beta_{j,b} := \prod_{\iota(i)=j} \alpha_{i,b} \left(= \prod_{\iota(i)=j} \alpha'_{i,b}\right). \tag{19}$$

Conclude the randomization step. Form the randomized matrices by

$$\hat{\mathbf{S}}_{i,b} := \mathrm{diag}\left(\alpha_{i,b}\mathbf{K}_{i-1}^{-1}\begin{pmatrix}\mathbf{M}_{i,b} & \\ & \mathbf{R}_{i,b}\end{pmatrix}\mathbf{K}_i\ ,\ \ \alpha'_{i,b}\mathbf{K'}_{i-1}^{-1}\begin{pmatrix}\mathbf{M'}_{i,b} & \\ & \mathbf{R}'_{i,b}\end{pmatrix}\mathbf{K'}_i\right) \in R^{t\times t} \tag{20}$$

2. Graded encoding. Apply graded encodings on the matrices $\hat{\mathbf{S}}_{i,b}$ and the bookends $\mathbf{J}, \mathbf{L}$.

For candidates that use GGH15 encoding (e.g. [GGH15, Section 5.2], [HHSS17]), following the notations defined in Construction 5.1, we compute

$$\begin{aligned}
&\mathbf{A}_J := \mathbf{J}\mathbf{A}_0 \in \mathbb{Z}^{1\times m}\\
&\mathbf{D}_{i,b} \leftarrow \mathbf{A}_{i-1}^{-1}(\hat{\mathbf{S}}_{i,b}\cdot\mathbf{A}_i + \mathbf{E}_{i,b}) \in \mathbb{Z}^{m\times m}, i = 1,\ldots,h-1\\
&\mathbf{D}_{h,b} \leftarrow \mathbf{A}_{h-1}^{-1}(\hat{\mathbf{S}}_{h,b}\cdot\mathbf{L}\cdot\mathbf{A}_h + \mathbf{E}_{h,b}) \in \mathbb{Z}^{m\times 1} \text{ where } \mathbf{A}_h \leftarrow U(\mathbb{Z}_q)
\end{aligned} \tag{21}$$

The obfuscated code consists of $\mathbf{A}_J, \{\mathbf{D}_{i,b}\}_{i\in[h],b\in\{0,1\}}$.

Remark 9.1 (merging the dummy program)**.** In the iO candidates like [GGH+13b, GGH15, HHSS17] that rely on a dummy program, the constructions typically use GGH15 encoding twice (with correlated matrices and appropriate bookends), once to encode the original program and a second to encode the dummy program. These constructions can be captured by our framework above by tweaking the distributions of $\mathbf{A}_1,\ldots,\mathbf{A}_{h-1}$ (to be block diagonal matrices) and $\{\mathbf{D}_{i,b}\}_{i\in[h],b\in\{0,1\}}$. Our rank attack also applies there, with the analogous heuristic assumptions.

9.2 Summary of the applicability of the (old and new) attacks

We describe new attacks against candidates using GGH15 and GGH13 graded encodings, even when all the "safeguards" (bundling scalars, Kilian, random diagonal padding) are included. The running time of the attack is polynomial for input-partitioned BPs (in particular read-once BPs). But for c-input-repeating BPs, the running time grows exponentially in c. This feature matches the best known attack [CLLT17] on iO candidates that use CLT13 graded encoding [CLT13].

Attacks	Type of MBP	Diagonal padding	S & K	Graded encodings
[CGH+15]	Input-partitioned	Yes	Yes	CLT13
[CLLT17]	Single-input, exp.(rep.)	Yes	Yes	CLT13
[MSZ16]	Special	No	Yes	GGH13
[ADGM17]	Single-input, exp.(rep.)	No	Yes	GGH13
[CGH17]	Permutation, input-partitioned	Yes	Yes	GGH13 & GGH15
This work	Single-input, exp.(rep.)	Yes	Yes	GGH13 & GGH15

Summary of the applicability of the attacks to candidate BP obfuscators.

Here "exp.(rep.)" means the cost of the attack grows exponentially over the repetition times of an input in the branching program. "S & K" stands for "Scalar & Kilian randomization".

To clarify the status, we mention a few existing iO candidates for branching programs that we do not know how to break. Current, even if all the safeguards (randomization mechanisms) are included in the candidates, the only chance to prevent the attack seems to "hack" the input-step pattern in the plaintext branching program. The existing solutions include (non exclusively):

1. Let the input bits repeat $O(\lambda)$ many times. In this case the current analysis indicates that our attack runs in time exponential in λ.
2. Converting the plaintext branching programs into dual-input BPs, as was used in [GMM+16].
3. Adding a "stamp function" to disturb the BP step pattern, as was used in [FRS17].

The status holds not only for the candidates using GGH15, but also for those using GGH13 and CLT13 (at least against classical polynomial algorithms).

9.3 A distinguishing attack for iO candidates using GGH15

To break a candidate indistinguishability obfuscator for branching programs, it suffices to show a pair of branching programs $\Gamma^{(0)}, \Gamma^{(1)}$ that are functionally equivalent, yet distinguishable when they are obfuscated. We present such a distinguishing algorithm. Readers who prefer to understand the attack with concrete examples can turn to the later pages and find Examples 9.1 and 9.2, load them in the cache, then read the algorithm and the analysis with the examples in mind.

Algorithm 9.2 (rank attack)**.** The algorithm takes as input two plaintext branching programs $\Gamma^{(0)}, \Gamma^{(1)}$, the obfuscated code $\left(\mathbf{A}_J, \{\mathbf{D}_{i,b}\}_{i\in[h],b\in\{0,1\}}\right)$ of either $\Gamma^{(0)}$ or $\Gamma^{(1)}$, proceeds as follows.

1. Pick two integers $\rho \geq T \geq 1$ according to $\Gamma^{(0)}, \Gamma^{(1)}$ and the security parameter.
2. Divide the ℓ input bits into 2 intervals $[\ell] = \mathcal{X} \mid \mathcal{Y}$ such that $|\mathcal{X}|, |\mathcal{Y}| \geq \lceil \log \rho \rceil$.
3. For $1 \leq i, j \leq \rho$, evaluate the obfuscated code on ρ^2 different inputs of the form $u^{(i,j)} = x^{(i)} \mid y^{(j)} \in \{0,1\}^\ell$ s.t. $f(u^{(i,j)}) = 0$. The choice of the inputs depends on $\Gamma^{(0)}, \Gamma^{(1)}$ and we will explain later. Let $v^{(i,j)} \in \mathbb{Z}$ be the evaluation result on $u^{(i,j)}$:
$$v^{(i,j)} := \mathbf{A}_J \cdot \mathbf{D}_{\varpi(u^{(i,j)})}.$$
4. Compute the rank of matrix $\mathbf{V} = (v^{(i,j)}) \in \mathbb{Z}^{\rho\times\rho}$, then compare the rank with the threshold T to decide whether the obfuscated program is $\Gamma^{(0)}$ or $\Gamma^{(1)}$.

The values of T and ρ, and also the ρ^2 inputs depend on the details of $\Gamma^{(0)}, \Gamma^{(1)}$. We will determine these values and inputs in the analyses. The total cost of the attack is $\mathrm{poly}(\rho)$.

9.3.1 Analysis of the rank attack on read-once branching programs

We analyze Algorithm 9.2 on the obfuscated read-once branching programs (which means $h = \ell$, ϖ is an identity function.)

Remark on the candidate obfuscator of Halevi et al. The candidate obfuscator of Halevi et al. [HHSS17] is the only one in the literature that explicitly works with read-once branching programs. To "keep within the realm of feasibility", [HHSS17] removes some of the safeguards. In particular the bundling scalars are not added in [HHSS17] since they only consider read-once branching programs. Also Kilian-style randomization are not included since the $\mathbf{A}$ matrices in GGH15 encoding already provide this kind of protection. They set $\mathcal{P}_0 = \{\mathbf{0}^{w\times w}\}$. The dummy branch and the bookends are set correspondingly to detect whether $\mathbf{M}_x = \mathbf{0}^{w\times w}$. Concretely $\hat{\mathbf{S}}_{i,b} = \mathrm{diag}\,(\mathbf{M}_{i,b}, \mathbf{R}_{i,b}, \mathbf{I}^{w\times w}, \mathbf{R}_{i,b})$, $\mathbf{J} = (\mathbf{1}^{1\times w} \mid \mathbf{1}^{1\times n} \mid \mathbf{0}^{1\times w} \mid -\mathbf{1}^{1\times n})$, $\mathbf{L} = \mathbf{1}^{w+n+w+n}$. Since all the entries of $\mathbf{M}$ are non-negative, then $\mathbf{1}^{1\times w} \cdot \mathbf{P}_1 \cdot \mathbf{1}^w = 0$ over $\mathbb{Z}_q$ means $\mathbf{P}_1 = \mathbf{0}$.

The subtraction attack. We note that without the bundling factors, the obfuscated code of $\Gamma^{(0)}$ and $\Gamma^{(1)}$ might be distinguished simply from illegal evaluations even if they are read-once and functionally equivalent. The idea is to compute $\mathbf{D}_{i,0} - \mathbf{D}_{i,1}$ which gives the encoding of

$$\hat{\mathbf{S}}_{i,0} - \hat{\mathbf{S}}_{i,1} = \text{diag}\left(\mathbf{M}_{i,0} - \mathbf{M}_{i,1}, \mathbf{R}_{i,0} - \mathbf{R}_{i,1}, \mathbf{0}^{w\times w}, \mathbf{R}_{i,0} - \mathbf{R}_{i,1}\right).$$

This may leak extra information beyond the honest evaluations. As a concrete example, consider the following pairs of 1-step branching programs.

$$\Gamma^{(0)} = \left\{\mathbf{M}_{1,0} = \begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix}, \mathbf{M}_{1,1} = \begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix}\right\}, \Gamma^{(1)} = \left\{\mathbf{M}_{1,0} = \begin{pmatrix}1 & 0\\ 0 & 0\end{pmatrix}, \mathbf{M}_{1,1} = \begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix}\right\}$$

Both programs compute the 1-bit functionality $f(x) = 1$ so they are functionally equivalent. But the distinguisher can simply compute $\mathbf{D}_{1,0} - \mathbf{D}_{1,1}$ and check if it encodes an all-0 matrix or not.

GGH15 obfuscator for read-once branching programs with all the safeguards. Instead of [HHSS17] we analyze a variant of the candidate from [GGH15, Section 5.2] where we apply all the known safeguard mechanisms. For the ease of presentation we make the following specification on the type of read-once branching programs we are working with.

Definition 13. Let $\mathbf{e}_1 \in \{0,1\}^{1\times w}$ denotes the unit vector with the 1^{st} coordinate being 1, the rest being 0. We require a read-once branching program $\Gamma = \left\{\{\mathbf{M}_{i,b} \in \{0,1\}^{w\times w}\}_{i\in[h],b\in\{0,1\}}, \mathcal{P}_0, \mathcal{P}_1\right\}$ to satisfy $\mathbf{e}_1 \cdot \mathcal{P}_0 \cdot \mathbf{e}_1^T = \{1\}$, $\mathbf{e}_1 \cdot \mathcal{P}_1 \cdot \mathbf{e}_1^T = \{0\}$.

Construction 9.3 (iO candidate for read-once BPs with all the safeguards)**.** Given a BP Γ that satisfies Def. 13. Let $t = 2(wn+n)$. Sample $\left\{\mathbf{S}_{i,b}, \mathbf{R}_{i,b}, \mathbf{R}'_{i,b} \in \mathbb{Z}^{n\times n}\right\}_{i\in[h],b\in\{0,1\}}$, $\left\{\mathbf{K}_i, \mathbf{K}_i^{-1}, \mathbf{K'}_i, \mathbf{K'}_i^{-1} \in \mathbb{Z}^{t\times t}\right\}_{i\in[h-1]}$, $\mathbf{K}_0 = \mathbf{K}_h = \mathbf{K'}_0 = \mathbf{K'}_h = \mathbf{I}^{t/2\times t/2}$, $\mathbf{r}_J, \mathbf{r}_L \in \mathbb{Z}^{1\times w}$, all randomly with small norm. Set $\mathbf{J} = (\mathbf{e}_1 \otimes \mathbf{r}_J \mid \mathbf{0}^{1\times n} \mid -\mathbf{e}_1 \otimes \mathbf{r}_J \mid \mathbf{0}^{1\times n})$, $\mathbf{L} = (\mathbf{e}_1 \otimes \mathbf{r}_L \mid \mathbf{1}^{1\times n} \mid \mathbf{e}_1 \otimes \mathbf{r}_L \mid \mathbf{1}^{1\times n})^T$.

Compute $\left\{\hat{\mathbf{S}}_{i,b} \in \mathbb{Z}^{t\times t}\right\}_{i\in[h],b\in\{0,1\}}$ as

$$\hat{\mathbf{S}}_{i,b} := \text{diag}\left(\mathbf{K}_{i-1}^{-1}\begin{pmatrix}\mathbf{M}_{i,b}\otimes\mathbf{S}_{i,b} & \\ & \mathbf{R}_{i,b}\end{pmatrix}\mathbf{K}_i\ ,\ \mathbf{K'}_{i-1}^{-1}\begin{pmatrix}\mathbf{I}^{w\times w}\otimes\mathbf{S}_{i,b} & \\ & \mathbf{R}'_{i,b}\end{pmatrix}\mathbf{K'}_i\right). \tag{22}$$

The obfuscated code is the encoding of $\left(\mathbf{J}, \left\{\hat{\mathbf{S}}_{i,b}\right\}_{i\in[h],b\in\{0,1\}}, \mathbf{L}\right)$ produced same as Eqn. (21).

The difference of Eqn. (22) from Eqn. (20) is that the bundling factors are non-commutative. It does not break correctness since we are working with read-once branching programs.

Analysis of the rank attack. We show Algorithm 9.2 breaks Construction 9.3 (as well as the construction in [HHSS17]). Most of the analysis will be explained based on elementary observations that are *sufficient* for the attack. In the end we will extract the essential ingredients that are *necessary* in Remark 9.6.

In the analysis we make a few heuristic assumptions. All the heuristics are about the linear independence of vectors from products of Gaussians. The assumptions have been experimentally verified. Also, similar assumptions were used in [CLLT16, CGH17].

Claim 9.4 (structure of $\mathbf{V}$)**.** The matrix $\mathbf{V} \in \mathbb{Z}^{\rho\times\rho}$ obtained in Algorithm 9.2 can be factorized into

$$\mathbf{V} := (v^{(i,j)}) = \underbrace{(\mathbf{X}_1 \mid \mathbf{X}_2)}_{=:\mathbf{X}\in\mathbb{Z}^{\rho\times(m+t)}} \cdot \underbrace{\begin{pmatrix}\mathbf{Y}_1\\ \mathbf{Y}_2\end{pmatrix}}_{=:\mathbf{Y}\in\mathbb{Z}^{(m+t)\times\rho}} \tag{23}$$

where $\mathbf{X}_1 \in \mathbb{Z}^{\rho\times t}$, $\mathbf{X}_2 \in \mathbb{Z}^{\rho\times m}$, $\mathbf{Y}_1 \in \mathbb{Z}^{t\times\rho}$, $\mathbf{Y}_2 \in \mathbb{Z}^{m\times\rho}$. With overwhelming probability, $\mathbf{X}_2$, $\mathbf{Y}$ are full-rank over $\mathbb{Q}$ (i.e. their ranks are independent of the underlying branching program Γ); the intersection of the column spans of $\mathbf{X}_1$ and $\mathbf{X}_2$ is $\{\mathbf{0}\}$. In particular, when $\rho = t + m$, with overwhelming probability, $\mathsf{rank}(\mathbf{X}_1) = \mathsf{rank}(\mathbf{V}) - m$. Both statements are based on heuristic assumptions.

Justification of Claim 9.4. The analysis of Claim 9.4 follows the similar observations from [Hal15, CLLT16, CGH17]. Note that even though the matrix $\mathbf{X}_2$ depends on the underlying $\left\{\hat{\mathbf{S}}_{i,b}\right\}_{i\in[h],b\in\{0,1\}}$ and the underlying branching program, we will argue heuristically that the rank of $\mathbf{X}_2$ does not; in fact, the rank of $\mathbf{X}_2$ is m. The same applies also to $\mathbf{Y}_1$ and $\mathbf{Y}_2$.

For concreteness we recap some of the details. Let the input partition be $[\ell] = [1,\nu] \mid [\nu+1,\ell]$ for a fixed integer ν. The honest evaluation of the obfuscated code on input $u^{(i,j)} = x^{(i)} \mid y^{(j)} \in \{0,1\}^{\nu} \mid \{0,1\}^{\ell-\nu}$ gives via (10)[6]:

$$\mathbf{A}_J \cdot \mathbf{D}_{u^{(i,j)}} = \left(\mathbf{J}\cdot\hat{\mathbf{S}}_{x^{(i)}} \mid \mathbf{J}\cdot\mathbf{H}_{x^{(i)}}\right)\cdot\begin{pmatrix}\hat{\mathbf{S}}_{y^{(j)}}\cdot\mathbf{L}\mathbf{A}_\ell + \mathbf{H}_{y^{(j)}}\\ \mathbf{D}_{y^{(j)}}\end{pmatrix} \pmod q, \tag{24}$$

where the subscripts do not strictly follow the convention of subset product. Instead, we define $\hat{\mathbf{S}}_{x^{(i)}} := \prod_{k=1}^{\nu}\left(\hat{\mathbf{S}}_{k,x_k^{(i)}}\right)$, $\hat{\mathbf{S}}_{y^{(j)}} := \prod_{k=1}^{\ell-\nu}\left(\hat{\mathbf{S}}_{k+\nu,y_k^{(j)}}\right)$, (the subscripts are not directly interpreted as bit-strings, e.g. even when $y^{(j)} = x^{(i)} = 000$, $\hat{\mathbf{S}}_{x^{(i)}}$ and $\hat{\mathbf{S}}_{y^{(j)}}$ denote different objects; same for below,) $\mathbf{D}_{y^{(j)}} := \prod_{k=1}^{\ell-\nu}\left(\mathbf{D}_{k+\nu,y_k^{(j)}}\right)$, the terms $\mathbf{H}_{x^{(i)}} \in \mathbb{Z}^{t\times m}$, $\mathbf{H}_{y^{(j)}} \in \mathbb{Z}^{t\times 1}$ are defined as[7]

$$\mathbf{H}_{x^{(i)}} := \sum_{k_2=1}^{\nu}\left(\prod_{k_1=1}^{k_2-1}\hat{\mathbf{S}}_{k_1,x_{k_1}^{(i)}}\cdot\mathbf{E}_{k_2,x_{k_2}^{(i)}}\cdot\prod_{k_3=k_2+1}^{\nu}\mathbf{D}_{k_3,x_{k_3}^{(i)}}\right), \tag{25}$$

$$\mathbf{H}_{y^{(j)}} := \sum_{k_2=1}^{\ell-\nu}\left(\prod_{k_1=1}^{k_2-1}\hat{\mathbf{S}}_{k_1+\nu,y_{k_1}^{(j)}}\cdot\mathbf{E}_{k_2+\nu,y_{k_2}^{(j)}}\cdot\prod_{k_3=k_2+1}^{\ell-\nu}\mathbf{D}_{k_3+\nu,y_{k_3}^{(j)}}\right). \tag{26}$$

[6]For example, when $\ell = 4$, $u = x \mid y = 00 \mid 00$, we have

$$\begin{aligned}\mathbf{A}_J\cdot\mathbf{D}_u &= \mathbf{J}(\hat{\mathbf{S}}_{1,0}\hat{\mathbf{S}}_{2,0}\hat{\mathbf{S}}_{3,0}\hat{\mathbf{S}}_{4,0}\mathbf{L}\mathbf{A}_4 + \hat{\mathbf{S}}_{1,0}\hat{\mathbf{S}}_{2,0}\hat{\mathbf{S}}_{3,0}\mathbf{E}_{4,0}\\ &\quad + \hat{\mathbf{S}}_{1,0}\hat{\mathbf{S}}_{2,0}\mathbf{E}_{3,0}\mathbf{D}_{4,0} + \hat{\mathbf{S}}_{1,0}\mathbf{E}_{2,0}\mathbf{D}_{3,0}\mathbf{D}_{4,0} + \mathbf{E}_{1,0}\mathbf{D}_{2,0}\mathbf{D}_{3,0}\mathbf{D}_{4,0})\\ &= (\mathbf{J}\hat{\mathbf{S}}_{1,0}\hat{\mathbf{S}}_{2,0} \mid \mathbf{J}(\hat{\mathbf{S}}_{1,0}\mathbf{E}_{2,0} + \mathbf{E}_{1,0}\mathbf{D}_{2,0}))\cdot\begin{pmatrix}\hat{\mathbf{S}}_{3,0}\hat{\mathbf{S}}_{4,0}\mathbf{L}\mathbf{A}_4 + \hat{\mathbf{S}}_{3,0}\mathbf{E}_{4,0} + \mathbf{E}_{3,0}\mathbf{D}_{4,0}\\ \mathbf{D}_{3,0}\mathbf{D}_{4,0}\end{pmatrix} \pmod q.\end{aligned}$$

[7] For example, when $\ell = 6$, $u = x \mid y = 000 \mid 000$, we have

$$\mathbf{H}_x = \hat{\mathbf{S}}_{1,0}\hat{\mathbf{S}}_{2,0}\mathbf{E}_{3,0} + \hat{\mathbf{S}}_{1,0}\mathbf{E}_{2,0}\mathbf{D}_{3,0} + \mathbf{E}_{1,0}\mathbf{D}_{2,0}\mathbf{D}_{3,0}, \mathbf{H}_y = \hat{\mathbf{S}}_{4,0}\hat{\mathbf{S}}_{5,0}\mathbf{E}_{6,0} + \hat{\mathbf{S}}_{4,0}\mathbf{E}_{5,0}\mathbf{D}_{6,0} + \mathbf{E}_{4,0}\mathbf{D}_{5,0}\mathbf{D}_{6,0}.$$

Recall that we assume for all $i, j \in [\rho]$, $f(u^{(i,j)}) = 0$, so $\mathbf{J} \cdot \hat{\mathbf{S}}_{u^{(i,j)}} \cdot \mathbf{L} = 0$ holds by the correctness of the construction. Hence we have

$$v^{(i,j)} = \left(\mathbf{J} \cdot \hat{\mathbf{S}}_{x^{(i)}} \mid \mathbf{J} \cdot \mathbf{H}_{x^{(i)}}\right) \cdot \begin{pmatrix} \mathbf{H}_{y^{(j)}} \\ \mathbf{D}_{y^{(j)}} \end{pmatrix} \in \mathbb{Z}. \tag{27}$$

Arrange $v^{(i,j)}$ into a matrix gives

$$\mathbf{V} := \begin{pmatrix} v^{(1,1)} & \dots & v^{(1,\rho)} \\ \dots & \dots & \dots \\ v^{(\rho,1)} & \dots & v^{(\rho,\rho)} \end{pmatrix} = \underbrace{\begin{pmatrix} \mathbf{J} \cdot \hat{\mathbf{S}}_{x^{(1)}} \mid \mathbf{J} \cdot \mathbf{H}_{x^{(1)}} \\ \dots \\ \mathbf{J} \cdot \hat{\mathbf{S}}_{x^{(\rho)}} \mid \mathbf{J} \cdot \mathbf{H}_{x^{(\rho)}} \end{pmatrix}}_{:=\left(\mathbf{X}_1 \mid \mathbf{X}_2\right) := \mathbf{X} \in \mathbb{Z}^{\rho \times (m+t)}} \cdot \underbrace{\begin{pmatrix} \mathbf{H}_{y^{(1)}} & \dots & \mathbf{H}_{y^{(\rho)}} \\ \mathbf{D}_{y^{(1)}} & & \mathbf{D}_{y^{(\rho)}} \end{pmatrix}}_{:= \begin{pmatrix} \mathbf{Y}_1 \\ \mathbf{Y}_2 \end{pmatrix} := \mathbf{Y} \in \mathbb{Z}^{(m+t) \times \rho}} \tag{28}$$

We first verify that $\mathbf{X}_2$ and $\mathbf{Y}$ are likely to be full-rank over $\mathbb{Q}$.

- For $\mathbf{Y}_2$, each column in $\mathbf{Y}_2$ is a subset product of $\mathbf{D}$ matrices (sampled from discrete-Gaussian) selected by a different input, each product contains at least a random column factor, which splits across the rows. This justifies the linear independence across the rows within $\mathbf{Y}_2$.
- For $\mathbf{Y}_1$, we look at the $\mathbf{F}_{y^{(j)}} := \mathbf{E}_{\nu+1,y_1^{(j)}} \cdot \prod_{k=2}^{\ell-\nu} \mathbf{D}_{\nu+k,y_k^{(j)}}$ term in $\mathbf{H}_{y^{(j)}}$ in Eqn. (26).If we write $\mathbf{Y}_1 = \mathbf{G} + \mathbf{F}$ where each column of $\mathbf{F}$ is $\mathbf{F}_{y^{(j)}}$, then the rows in $\mathbf{F}$ are likely to be linearly independent using the similar observation for $\mathbf{Y}_2$. The $\mathbf{G}$ component (which depends on the $\hat{\mathbf{S}}_{i,b}$'s), though not completely independent from $\mathbf{F}$, is not likely to "cancel $\mathbf{F}$ out".
- For $\mathbf{Y}$ as a whole, the rows in $\mathbf{Y}_1$, $\mathbf{Y}_2$ are likely to be linearly independent over $\mathbb{Q}$, since the $\mathbf{D}$ matrices in $\mathbf{Y}_2$ are sampled from Gaussian with a fixed center from the terms in $\mathbf{Y}_1$. For this reason $\mathbf{Y}_1$ and $\mathbf{Y}_2$ are inherently linearly dependent over $\mathbb{Z}_q$, as was mentioned in [Hal15] to emphasize that the attack must use equations hold over $\mathbb{Z}$.
- For the columns of $\mathbf{X}_2$, we look at the $\mathbf{E}_{1,x_1^{(i)}} \cdot \prod_{k=2}^{\nu} \mathbf{D}_{k,x_k^{(i)}}$ term in $\mathbf{H}_{x^{(i)}}$ in Eqn. (25), then use the similar observations from $\mathbf{Y}_1$ to argue that $\mathbf{X}_2$ is full-rank.

So when $\rho = t+m$, with overwhelming probability, $\mathbf{Y}$ is full-rank, i.e. $\mathsf{rank}(\mathbf{Y}) = m+t$; $\mathsf{rank}(\mathbf{X}_2) = m$. In particular the ranks of $\mathbf{X}_2$ and $\mathbf{Y}$ are system parameters and independent of the underlying matrix branching program Γ.

In addition, we claim that the intersection of the column spans of $\mathbf{X}_1$ and $\mathbf{X}_2$ is $\{\mathbf{0}\}$. The easiest way to justify is again to use the $\mathbf{E}_{1,x_1^{(i)}} \cdot \prod_{k=2}^{\nu} \mathbf{D}_{k,x_k^{(i)}}$ term in Eqn. (25) — the intersection of the column span of $\mathbf{X}_1$ and the column span of this term is likely to be $\{\mathbf{0}\}$.

To conclude, under all the heuristics about linear independence, $\mathsf{rank}(\mathbf{X}_1) = \mathsf{rank}(\mathbf{X}) - m = \mathsf{rank}(\mathbf{V}) - m$. Note that $\mathsf{rank}(\mathbf{X}_1)$ does depend on Γ and we will analyze it next.

Claim 9.5. There exists a pair of read-once branching programs that are functionally equivalent, yet Construction 9.3 produces $\mathbf{X}_1$ with different ranks, where $\mathbf{X}_1$ is defined in Claim 9.4.

Justification of Claim 9.5. First we examine the structure of $\mathbf{X}_1$:

$$\begin{aligned}\mathbf{X}_1 = \begin{pmatrix}\mathbf{J}\cdot\hat{\mathbf{S}}_{x^{(1)}}\\ \dots \\ \mathbf{J}\cdot\hat{\mathbf{S}}_{x^{(\rho)}}\end{pmatrix} &= \begin{pmatrix}\mathbf{J}\cdot\mathrm{diag}\left(\mathbf{M}_{x^{(1)}}\otimes\mathbf{S}_{x^{(1)}}, \mathbf{R}_{x^{(1)}}, \mathbf{I}^{w\times w}\otimes\mathbf{S}_{x^{(1)}}, \mathbf{R}'_{x^{(1)}}\right)\\ \dots \\ \mathbf{J}\cdot\mathrm{diag}\left(\mathbf{M}_{x^{(\rho)}}\otimes\mathbf{S}_{x^{(\rho)}}, \mathbf{R}_{x^{(\rho)}}, \mathbf{I}^{w\times w}\otimes\mathbf{S}_{x^{(\rho)}}, \mathbf{R}'_{x^{(\rho)}}\right)\end{pmatrix}\cdot\mathrm{diag}\left(\mathbf{K}_\nu, \mathbf{K'}_\nu\right)\\ &= \begin{pmatrix}(\mathbf{e}_1\mathbf{M}_{x^{(1)}})\otimes(\mathbf{r}_J\mathbf{S}_{x^{(1)}}) \mid \mathbf{0}^{1\times n} \mid -\mathbf{e}_1\otimes(\mathbf{r}_J\mathbf{S}_{x^{(1)}}) \mid \mathbf{0}^{1\times n}\\ \dots \\ (\mathbf{e}_1\mathbf{M}_{x^{(\rho)}})\otimes(\mathbf{r}_J\mathbf{S}_{x^{(\rho)}}) \mid \mathbf{0}^{1\times n} \mid -\mathbf{e}_1\otimes(\mathbf{r}_J\mathbf{S}_{x^{(\rho)}}) \mid \mathbf{0}^{1\times n}\end{pmatrix}\cdot\mathrm{diag}\left(\mathbf{K}_\nu, \mathbf{K'}_\nu\right)\end{aligned} \tag{29}$$

Since $\mathrm{diag}\left(\mathbf{K}_\nu, \mathbf{K'}_\nu\right)$ is always non-singular, rearranging the terms in Eqn. (29) gives

$$\mathsf{rank}(\mathbf{X}_1) = \mathsf{rank}\underbrace{\begin{pmatrix}(\mathbf{e}_1\mathbf{M}_{x^{(1)}} \mid -\mathbf{e}_1)\otimes(\mathbf{r}_J\mathbf{S}_{x^{(1)}})\\ \dots \\ (\mathbf{e}_1\mathbf{M}_{x^{(\rho)}} \mid -\mathbf{e}_1)\otimes(\mathbf{r}_J\mathbf{S}_{x^{(\rho)}})\end{pmatrix}}_{=:\mathbf{C}\in\mathbb{Z}^{(2wn)\times\rho}} \tag{30}$$

Take a closer look at the $\mathbf{C}$ matrix in Eqn. (30). First we observe that the rows $\mathbf{r}_J\mathbf{S}_{x^{(1)}}$ through $\mathbf{r}_J\mathbf{S}_{x^{(\rho)}}$ are likely to be linearly independent (for the similar reasons as $\mathbf{X}_2$, $\mathbf{Y}_1$, $\mathbf{Y}_2$). So we can think of $\mathbf{C}$ as being partitioned into $(2w)\times\rho$ blocks, each block contains an n-dimensional row vector, which is either $\mathbf{0}$ or random (across rows), depending on the entries in

$$\begin{pmatrix}\mathbf{e}_1\mathbf{M}_{x^{(1)}} \mid -\mathbf{e}_1\\ \dots \\ \mathbf{e}_1\mathbf{M}_{x^{(\rho)}} \mid -\mathbf{e}_1\end{pmatrix} =: \mathbf{B}\in\{0,1\}^{(2w)\times\rho} \tag{31}$$

For example, if $\mathbf{C} = \begin{pmatrix}\mathbf{r}_J\mathbf{S}_{x^{(1)}} & \mathbf{0} & -\mathbf{r}_J\mathbf{S}_{x^{(1)}} & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & -\mathbf{r}_J\mathbf{S}_{x^{(2)}} & \mathbf{0}\end{pmatrix}$, then $\mathbf{B} = \begin{pmatrix}1 & 0 & -1 & 0\\ 0 & 0 & -1 & 0\end{pmatrix}$.

Now we choose two branching programs $\Gamma^{(0)}$, $\Gamma^{(1)}$ so that the resulting $\mathbf{B}$ matrices, denoted as $\mathbf{B}^{(0)}$ and $\mathbf{B}^{(1)}$, have different ranks; therefore the corresponding $\mathbf{C}^{(0)}$, $\mathbf{C}^{(1)}$ (hence $\mathbf{X}_1^{(0)}$, $\mathbf{X}_1^{(1)}$) matrices have different ranks.

Example 9.1. Define the target sets as $\mathcal{P}_0 = \{\mathbf{Z}\in\{0,1\}^{w\times w}$ s.t. $\mathbf{Z}_{1,1} = 1\}$, $\mathcal{P}_1 = \{\mathbf{Z}\in\{0,1\}^{w\times w}$ s.t. $\mathbf{Z}_{1,1} = 0\}$. $\Gamma^{(0)}$ simply sets all the matrices to be identity, i.e.

$$\Gamma^{(0)} = \left\{\left\{\mathbf{M}^{(0)}_{i,b} = \mathbf{I}^{w\times w}\right\}_{i\in[\ell],b\in\{0,1\}}, \mathcal{P}_0, \mathcal{P}_1\right\};$$

$\Gamma^{(1)}$ is almost the same with $\Gamma^{(0)}$ except that $\mathbf{M}^{(1)}_{1,1} = \begin{pmatrix}1 & \mathbf{1}^{1\times(w-1)}\\ \mathbf{0}^{(w-1)\times 1} & \mathbf{I}^{(w-1)\times(w-1)}\end{pmatrix}$, i.e.

$$\Gamma^{(1)} = \left\{\mathbf{M}^{(1)}_{1,0} = \mathbf{I}^{w\times w}, \mathbf{M}^{(1)}_{1,1}, \left\{\mathbf{M}^{(1)}_{i,b} = \mathbf{I}^{w\times w}\right\}_{i\in[2,\ell],b\in\{0,1\}}, \mathcal{P}_0, \mathcal{P}_1\right\}.$$

Following the specification from Def. 13, both $\Gamma^{(0)}$, $\Gamma^{(1)}$ represent all-0 functionality.

The ρ^2 inputs are chosen as follows. For the prefixes $\left\{x^{(i)}\right\}_{i\in[\rho]}$, let $x^{(1)} = 1 \mid 0^{\nu-1}$. Let the rest be arbitrary from $0 \mid \{0,1\}^{\nu-1}$. The suffixes $\left\{y^{(j)}\right\}_{j\in[\rho]}$ are arbitrary from $\{0,1\}^{\ell-\nu}$.

Here is a comparison of the resulting $\mathbf{B}$ matrices

$$\mathbf{B}^{(0)} = \begin{pmatrix} \mathbf{e}_1\mathbf{M}^{(0)}_{x^{(1)}} \mid -\mathbf{e}_1 \\ \mathbf{e}_1\mathbf{M}^{(0)}_{x^{(2)}} \mid -\mathbf{e}_1 \\ \cdots \\ \mathbf{e}_1\mathbf{M}^{(0)}_{x^{(\rho)}} \mid -\mathbf{e}_1 \end{pmatrix} = \begin{pmatrix} \mathbf{e}_1 \mid -\mathbf{e}_1 \\ \mathbf{e}_1 \mid -\mathbf{e}_1 \\ \cdots \\ \mathbf{e}_1 \mid -\mathbf{e}_1 \end{pmatrix}, \quad \mathbf{B}^{(1)} = \begin{pmatrix} \mathbf{e}_1\mathbf{M}^{(1)}_{x^{(1)}} \mid -\mathbf{e}_1 \\ \mathbf{e}_1\mathbf{M}^{(1)}_{x^{(2)}} \mid -\mathbf{e}_1 \\ \cdots \\ \mathbf{e}_1\mathbf{M}^{(1)}_{x^{(\rho)}} \mid -\mathbf{e}_1 \end{pmatrix} = \begin{pmatrix} \mathbf{1}^{1\times w} \mid -\mathbf{e}_1 \\ \mathbf{e}_1 \mid -\mathbf{e}_1 \\ \cdots \mid \cdots \\ \mathbf{e}_1 \mid -\mathbf{e}_1 \end{pmatrix} \tag{32}$$

So $\mathsf{rank}(\mathbf{B}^{(0)}) = 1$, $\mathsf{rank}(\mathbf{B}^{(1)}) = 2$. Therefore w.h.p. (over the randomness of $\mathbf{r}$ and the $\mathbf{S}$ matrices) $\mathsf{rank}(\mathbf{C}^{(0)}) = n$, $\mathsf{rank}(\mathbf{C}^{(1)}) = n+1$. Hence $\mathsf{rank}(\mathbf{X}_1^{(0)}) = n$, $\mathsf{rank}(\mathbf{X}_1^{(1)}) = n+1$.

Combining Claim 9.4 and Claim 9.5, we conclude that there exists two functionally equivalent BPs $\Gamma^{(0)}$, $\Gamma^{(1)}$ whose obfuscated forms (based on Construction 9.3) are distinguishable by Algorithm 9.2 by setting $\rho = m + t = m + 2(nw + n)$, $T = m + n$. Concretely, if $\mathsf{rank}(\mathbf{V}) \leq T$, the distinguisher chooses $\Gamma^{(0)}$; if $\mathsf{rank}(\mathbf{V}) > T$, the distinguisher chooses $\Gamma^{(1)}$. The success probability is close to 1 under the heuristics about linear independence. The cost of Algorithm 9.2 is polynomial in ρ, hence polynomial in the input length ℓ and the security parameter.

Remark 9.6 (Weakened conditions for the success of the attack)**.** The threshold rank T may be different, depending on the plaintext branching programs.

The conditions of the matrices in of Eqn. (28) can be weakened by requiring

- The columns of $\mathbf{X}_1$ do not fall into the column span of $\mathbf{X}_2$;
- There are sufficiently many rows in $\mathbf{Y}_1$ that do not fall into the row span of $\mathbf{Y}_2$;

instead of requiring the entire $\mathbf{X}_2$ and $\mathbf{Y}$ to be full-rank. Under the weakened requirements, the rank of $\mathbf{X}_1$ can still be extracted, since it is multiplied with an (at least partially) random matrix $\mathbf{Y}_1$, plus the fact that neither $\mathbf{X}_1$ nor $\mathbf{Y}_1$ fall into the span of $\mathbf{X}_2$ and $\mathbf{Y}_2$. Indeed there might be ways to modify the distributions of the random matrices so that $\mathbf{X}_2$ and $\mathbf{Y}$ are not full-rank, but we do not see how to bypass the weakened conditions without trivially breaking the candidate.

Remark 9.7 (Applicability of the analysis)**.** The analysis directly applies to break general $(2, \log(\rho))$-input-partitioned branching programs, not only read-once BPs. We need at least $\log(\rho)$ free bits in each interval to produce ρ differing prefixes and suffixes.

9.3.2 Analysis of the attack on general input-repeating branching programs

For general branching programs that are c-input-repeating, Algorithm 9.2 works as-it-is, except that the dimension ρ of the matrix $\mathbf{V}$ grows exponentially in c. We refer to this attack as "advanced zeroizing attack", meaning that the algorithm still works on evaluations of zeros, exploits the rank of some matrix $\mathbf{V}$, but uses more complicated analyses.

Additional background on tensor product. We recall some equations from [CLLT17, ADGM17] that are useful for switching the order of matrix productions.

For a matrix $\mathbf{A} = (\mathbf{a}_1 \mid ... \mid \mathbf{a}_m) \in \mathbb{R}^{n\times m}$, let $\mathsf{vec}(\mathbf{A}) = (\mathbf{a}_1^T \mid ... \mid \mathbf{a}_m^T)^T \in \mathbb{R}^{(nm)\times 1}$ denote the vectorization of $\mathbf{A}$ by stacking the columns one over another. For matrices $\mathbf{B}, \mathbf{C}, \mathbf{D}, \mathbf{E}, \mathbf{F}$ with proper dimensions, we have

$$\mathsf{vec}(\mathbf{B} \cdot \mathbf{C} \cdot \mathbf{D}) = (\mathbf{D}^T \otimes \mathbf{B}) \cdot \mathsf{vec}(\mathbf{C}) \tag{33}$$

Let $\mathbf{G}$ be a column vector, the dimension of $\mathbf{E}$ be $d_1 \times d_2$, the following equation holds:

$$\mathsf{vec}(\mathbf{B}\cdot\mathbf{C}\cdot\mathbf{D}\cdot\mathbf{E}\cdot\mathbf{F}\cdot\mathbf{G}) = \left(\mathsf{vec}(\mathbf{E})^T\otimes\mathbf{B}\right)\cdot\left(\mathbf{F}\otimes\mathbf{I}^{d_1\times d_1}\otimes\mathbf{C}\right)\cdot\mathsf{vec}\left(\mathbf{G}^T\otimes\mathbf{D}\right) \tag{34}$$

The analysis for 2-input-repeating branching programs. We start by analyzing Algorithm 9.2 on 2-input-repeating branching programs. For simplicity we assume $h = 2\ell$, the input bit sequence is "$1, 2, ..., \ell, 1, 2, ..., \ell$", i.e. $\varpi(x) = x \mid x$. The obfuscation scheme from Section 9.1 uses commutative bundling scalars, but we will not take advantage of the commutativity. The original scheme from [GGH15, Section 5.2] uses $R = \mathbb{Z}[X]/(\phi(X))$ as the base ring. Here we assume the base ring is $\mathbb{Z}$ and again we will not take advantage of the discrepancy.

The structure of $\mathbf{V}$. Recall from Algorithm 9.2 that the ℓ input bits are partitioned into two intervals $[\ell] = \mathcal{X} \mid \mathcal{Y} = [1,\nu] \mid [\nu+1,\ell]$, let the corresponding BP steps be partitioned into 4 intervals $[h] = \mathcal{X}_1 \mid \mathcal{Y}_1 \mid \mathcal{X}_2 \mid \mathcal{Y}_2 = [1,\nu] \mid [\nu+1,\ell] \mid [\ell+1,\ell+\nu] \mid [\ell+\nu+1,2\ell]$, where the steps in $\mathcal{X}_1, \mathcal{X}_2$ are controlled by input bits in $\mathcal{X}$, the steps in $\mathcal{Y}_1, \mathcal{Y}_2$ are controlled by input bits in $\mathcal{Y}$. Denote $\varpi(u^{(i,j)}) = \varpi(x^{(i)} \mid y^{(j)}) =: x_1^{(i)} \mid y_1^{(j)} \mid x_2^{(i)} \mid y_2^{(j)}$.

Express $v^{(i,j)} \in \mathbb{Z}$ as the evaluation result on input $u^{(i,j)} = x^{(i)} \mid y^{(j)}$ s.t. $f(u^{(i,j)}) = 0$:

$$\begin{aligned}
&\mathbf{A}_J\cdot\mathbf{D}_{\varpi(u^{(i,j)})}\\
=\ &\mathbf{A}_J\cdot\mathbf{D}_{x_1^{(i)}}\cdot\mathbf{D}_{y_1^{(j)}}\cdot\mathbf{D}_{x_2^{(i)}}\cdot\mathbf{D}_{y_2^{(j)}}\\
=\ &\mathbf{J}\hat{\mathbf{S}}_{x_1^{(i)}}\hat{\mathbf{S}}_{y_1^{(j)}}\hat{\mathbf{S}}_{x_2^{(i)}}\mathbf{H}_{y_2^{(j)}} + \mathbf{J}\hat{\mathbf{S}}_{x_1^{(i)}}\hat{\mathbf{S}}_{y_1^{(j)}}\mathbf{H}_{x_2^{(i)}}\mathbf{D}_{y_2^{(j)}} + \mathbf{J}\hat{\mathbf{S}}_{x_1^{(i)}}\mathbf{H}_{y_1^{(j)}}\mathbf{D}_{x_2^{(i)}}\mathbf{D}_{y_2^{(j)}} + \mathbf{J}\mathbf{H}_{x_1^{(i)}}\mathbf{D}_{y_1^{(j)}}\mathbf{D}_{x_2^{(i)}}\mathbf{D}_{y_2^{(j)}}\\
=\ &\left(\mathsf{vec}(\hat{\mathbf{S}}_{x_2^{(i)}})^T\otimes\left(\mathbf{J}\hat{\mathbf{S}}_{x_1^{(i)}}\right)\right)\cdot\mathsf{vec}\left({\mathbf{H}_{y_2^{(j)}}}^T\otimes\hat{\mathbf{S}}_{y_1^{(j)}}\right) + \left(\mathsf{vec}(\mathbf{H}_{x_2^{(i)}})^T\otimes\left(\mathbf{J}\hat{\mathbf{S}}_{x_1^{(i)}}\right)\right)\cdot\mathsf{vec}\left({\mathbf{D}_{y_2^{(j)}}}^T\otimes\hat{\mathbf{S}}_{y_1^{(j)}}\right)\\
&+\left(\mathsf{vec}(\mathbf{D}_{x_2^{(i)}})^T\otimes\left(\mathbf{J}\hat{\mathbf{S}}_{x_1^{(i)}}\right)\right)\cdot\mathsf{vec}\left({\mathbf{D}_{y_2^{(j)}}}^T\otimes\mathbf{H}_{y_1^{(j)}}\right) + \left(\mathsf{vec}(\mathbf{D}_{x_2^{(i)}})^T\otimes\left(\mathbf{J}\mathbf{H}_{x_1^{(i)}}\right)\right)\cdot\mathsf{vec}\left({\mathbf{D}_{y_2^{(j)}}}^T\otimes\mathbf{D}_{y_1^{(j)}}\right)
\end{aligned} \tag{35}$$

where the meanings of subscripts of $\hat{\mathbf{S}}, \mathbf{D}, \mathbf{H}$ are defined analogously as was in Eqn. (24) in the read-once setting. The third equality follows Formula (34) by setting the $\mathbf{C}$, $\mathbf{F}$ components in Formula (34) to be identity matrices with the correct dimensions.

Expressing Eqn. (35) as the inner product of two vectors (like what was done in Eqn. (27)) gives

$$\underbrace{\left(\mathsf{vec}(\hat{\mathbf{S}}_{x_2^{(i)}})^T\otimes\left(\mathbf{J}\hat{\mathbf{S}}_{x_1^{(i)}}\right) \mid \mathsf{vec}(\mathbf{H}_{x_2^{(i)}})^T\otimes\left(\mathbf{J}\hat{\mathbf{S}}_{x_1^{(i)}}\right) \mid \mathsf{vec}(\mathbf{D}_{x_2^{(i)}})^T\otimes\left(\mathbf{J}\hat{\mathbf{S}}_{x_1^{(i)}}\right) \mid \mathsf{vec}(\mathbf{D}_{x_2^{(i)}})^T\otimes\left(\mathbf{J}\mathbf{H}_{x_1^{(i)}}\right)\right)}_{:=\mathbf{x}^{(i)}}$$
$$\cdot\underbrace{\begin{pmatrix}\mathsf{vec}\left({\mathbf{H}_{y_2^{(j)}}}^T\otimes\hat{\mathbf{S}}_{y_1^{(j)}}\right)\\ \mathsf{vec}\left({\mathbf{D}_{y_2^{(j)}}}^T\otimes\hat{\mathbf{S}}_{y_1^{(j)}}\right)\\ \mathsf{vec}\left({\mathbf{D}_{y_2^{(j)}}}^T\otimes\mathbf{H}_{y_1^{(j)}}\right)\\ \mathsf{vec}\left({\mathbf{D}_{y_2^{(j)}}}^T\otimes\mathbf{D}_{y_1^{(j)}}\right)\end{pmatrix}}_{:=\mathbf{y}^{(j)}} \tag{36}$$

The dimensions of the 4 sectors in the $\mathbf{x}^{(i)}$ vector are, from left to right, t^3, $t^2\cdot m$, $t\cdot m^2$, m^3. So as for $\mathbf{y}^{(j)}$ from top to bottom. Let $\rho \geq t^3 + t^2\cdot m + t\cdot m^2 + m^3$.

The matrix $\mathbf{V} = (v^{(i,j)})$ can be expressed as

$$\mathbf{V} := \begin{pmatrix} v^{(1,1)} & \dots & v^{(1,\rho)} \\ \dots & \dots & \dots \\ v^{(\rho,1)} & \dots & v^{(\rho,\rho)} \end{pmatrix} = \begin{pmatrix} \dots \\ \mathbf{x}^{(i)} \\ \dots \end{pmatrix} \cdot \left(\dots \,\middle|\, \mathbf{y}^{(j)} \,\middle|\, \dots \right) =: \left(\mathbf{X}_1 \,\middle|\, \mathbf{X}_2 \,\middle|\, \mathbf{X}_3 \,\middle|\, \mathbf{X}_4 \right) \cdot \begin{pmatrix} \mathbf{Y}_1 \\ \mathbf{Y}_2 \\ \mathbf{Y}_3 \\ \mathbf{Y}_4 \end{pmatrix} \quad (37)$$

where $\mathbf{X}_1 = \begin{pmatrix} \dots \\ \mathsf{vec}(\hat{\mathbf{S}}_{x_2^{(i)}})^T \otimes \left(\mathbf{J}\hat{\mathbf{S}}_{x_1^{(i)}} \right) \\ \dots \end{pmatrix} \in \mathbb{Z}^{\rho \times (t^3)}$, $\mathbf{Y}_1 = \left(\dots \,\middle|\, \mathsf{vec}\left({\mathbf{H}_{y_2^{(j)}}}^T \otimes \hat{\mathbf{S}}_{y_1^{(j)}} \right) \,\middle|\, \dots \right) \in \mathbb{Z}^{(t^3) \times \rho}$;
analogously for $\mathbf{X}_2, \mathbf{X}_3, \mathbf{X}_4, \mathbf{Y}_2, \mathbf{Y}_3, \mathbf{Y}_4$.

Intermezzo. We can of course continue analyzing the ranks of every piece of $\mathbf{X}_1, ..., \mathbf{X}_4, \mathbf{Y}_1, ..., \mathbf{Y}_4$, and their dependency on the underlying branching program Γ, like what we did in the read-once setting; but it will constitute a significant blowup in the length of this article.

Instead, we follow the analysis strategy mentioned in Remark 9.6. First observe that (w.h.p. under the linear independence heuristics)

- The ranks of $\mathbf{Y}_3$, $\mathbf{X}_4$, $\mathbf{Y}_4$ are independent of the branching program Γ;
- The ranks of $\mathbf{X}_3$, $\mathbf{X}_1$, $\mathbf{Y}_1$, $\mathbf{X}_2$, $\mathbf{Y}_2$ depend on the branching program Γ.

So we claim it suffices to dig into the $\mathbf{X}_3 \cdot \mathbf{Y}_3$ component. Informally, the goal is to show (1) the "rank-gap" can be produced at $\mathbf{X}_3$; (2) the rank-gap will not be vanished in the other components so that it is detectable from $\mathbf{V}$.

Claim 9.8. There exists a pair of functionally equivalent 2-input-repeating BPs $\Gamma^{(0)}, \Gamma^{(1)}$ such that the following conditions hold with high probability,

1. Let $\Delta := \mathsf{rank}(\mathbf{X}_3^{(1)}) - \mathsf{rank}(\mathbf{X}_3^{(0)})$, then $\Delta > 0$;
2. $\mathsf{rank}(\mathbf{V}^{(1)}) - \mathsf{rank}(\mathbf{V}^{(0)}) \geq \Delta$.

where $\mathbf{X}_3^{(b)}, \mathbf{V}_3^{(b)}$ are defined in Eqn. (37) obtained from the obfuscation of $\Gamma^{(b)}$ using the construction in Section 9.1.

Similar to the read-once setting, all the heuristics are about linear independence.

Justification of Claim 9.8.1. We start from analyzing $\mathbf{X}_3, \mathbf{Y}_3$. Recall that

$$\mathbf{X}_3 = \begin{pmatrix} \dots \\ \mathsf{vec}(\mathbf{D}_{x_2^{(i)}})^T \otimes \left(\mathbf{J}\hat{\mathbf{S}}_{x_1^{(i)}} \right) \\ \dots \end{pmatrix} \in \mathbb{Z}^{\rho \times (t \cdot m^2)}, \quad \mathbf{Y}_3 = \left(\dots \,\middle|\, \mathsf{vec}\left({\mathbf{D}_{y_2^{(j)}}}^T \otimes \mathbf{H}_{y_1^{(j)}} \right) \,\middle|\, \dots \right) \in \mathbb{Z}^{(t \cdot m^2) \times \rho}; \quad (38)$$

We first claim $\mathbf{Y}_3$ is full-rank w.h.p., since the j^{th} column of $\mathbf{Y}_3$ is the vectorization of $\mathbf{D}_{y_2^{(j)}}^T \otimes \mathbf{H}_{y_1^{(j)}}$, which by itself is a tensor product of two random'ish matrices. The columns are likely to be linearly independent due to the different $\mathbf{D}_{y_2^{(j)}}$ and $\mathbf{H}_{y_1^{(j)}}$ across different js.

The rank of $\mathbf{X}_3$ though is Γ-dependent.

$$\begin{aligned}\mathbf{X}_3 &= \begin{pmatrix} \mathsf{vec}(\mathbf{D}_{x_2^{(1)}})^T \otimes \left(\mathbf{J} \cdot \mathrm{diag}\left(\alpha_{x^{(1)}}\mathbf{M}_{x^{(1)}}, \alpha_{x^{(1)}}\mathbf{R}_{x^{(1)}}, \alpha'_{x^{(1)}}\mathbf{I}^{w\times w}, \alpha'_{x^{(1)}}\mathbf{R}'_{x^{(1)}}\right) \cdot \mathrm{diag}\left(\mathbf{K}_\nu, \mathbf{K'}_\nu\right)\right) \\ \dots \\ \mathsf{vec}(\mathbf{D}_{x_2^{(\rho)}})^T \otimes \left(\mathbf{J} \cdot \mathrm{diag}\left(\alpha_{x^{(\rho)}}\mathbf{M}_{x^{(\rho)}}, \alpha_{x^{(\rho)}}\mathbf{R}_{x^{(\rho)}}, \alpha'_{x^{(\rho)}}\mathbf{I}^{w\times w}, \alpha'_{x^{(\rho)}}\mathbf{R}'_{x^{(\rho)}}\right) \cdot \mathrm{diag}\left(\mathbf{K}_\nu, \mathbf{K'}_\nu\right)\right) \end{pmatrix} \\ &= \begin{pmatrix} \mathsf{vec}(\mathbf{D}_{x_2^{(1)}})^T \otimes \left(\alpha_{x^{(1)}}\mathbf{r}_J\mathbf{M}_{x^{(1)}} \mid \mathbf{0}^{1\times n} \mid \alpha'_{x^{(1)}}\mathbf{r}'_J \mid \mathbf{0}^{1\times n}\right) \\ \dots \\ \mathsf{vec}(\mathbf{D}_{x_2^{(\rho)}})^T \otimes \left(\alpha_{x^{(\rho)}}\mathbf{r}_J\mathbf{M}_{x^{(\rho)}} \mid \mathbf{0}^{1\times n} \mid \alpha'_{x^{(\rho)}}\mathbf{r}'_J \mid \mathbf{0}^{1\times n}\right) \end{pmatrix} \cdot \left(\mathbf{I}^{m^2\times m^2} \otimes \mathrm{diag}\left(\mathbf{K}_\nu, \mathbf{K'}_\nu\right)\right)\end{aligned} \tag{39}$$

Since $\left(\mathbf{I}^{m^2\times m^2} \otimes \mathrm{diag}\left(\mathbf{K}_\nu, \mathbf{K'}_\nu\right)\right)$ is non-singular, we get

$$\mathsf{rank}(\mathbf{X}_3) = \mathsf{rank} \underbrace{\begin{pmatrix} \mathsf{vec}(\mathbf{D}_{x_2^{(1)}})^T \otimes \left(\alpha_{x^{(1)}}\mathbf{r}_J\mathbf{M}_{x^{(1)}} \mid \alpha'_{x^{(1)}}\mathbf{r}'_J\right) \\ \dots \\ \mathsf{vec}(\mathbf{D}_{x_2^{(\rho)}})^T \otimes \left(\alpha_{x^{(\rho)}}\mathbf{r}_J\mathbf{M}_{x^{(\rho)}} \mid \alpha'_{x^{(\rho)}}\mathbf{r}'_J\right) \end{pmatrix}}_{=:\mathbf{C}\in\mathbb{Z}^{\rho\times(m^2\cdot 2w)}} \tag{40}$$

The analysis of the $\mathbf{C}$ matrix in Eqn. (40) bears similarity to the analysis of Eqn. (30). First we observe that the rows $\mathsf{vec}\left(\mathbf{D}_{x_2^{(1)}}\right)^T$ through $\mathsf{vec}\left(\mathbf{D}_{x_2^{(\rho)}}\right)^T$ are random'ish and likely to be linearly independent. We therefore focus on the Γ-sensitive component which is

$$\mathbf{B} := \begin{pmatrix} \alpha_{x^{(1)}}\mathbf{r}_J\mathbf{M}_{x^{(1)}} \mid \alpha'_{x^{(1)}}\mathbf{r}'_J \\ \dots \\ \alpha_{x^{(\rho)}}\mathbf{r}_J\mathbf{M}_{x^{(\rho)}} \mid \alpha'_{x^{(\rho)}}\mathbf{r}'_J \end{pmatrix} \in \mathbb{Z}^{(2w)\times\rho} \tag{41}$$

Now we choose two branching programs $\Gamma^{(0)}$, $\Gamma^{(1)}$ so that the resulting $\mathbf{B}$ matrices, denoted as $\mathbf{B}^{(0)}$ and $\mathbf{B}^{(1)}$, have different ranks.

Example 9.2. Let $h = 2\ell$. Given the target sets $\mathcal{P}_0, \mathcal{P}_1$ specified in Eqn. (17), $\Gamma^{(0)}$ simply sets all the matrices to be identity, i.e.

$$\Gamma^{(0)} = \left\{\left\{\mathbf{M}^{(0)}_{i,b} = \mathbf{I}^{w\times w}\right\}_{i\in[h],b\in\{0,1\}}, \mathcal{P}_0, \mathcal{P}_1\right\};$$

Let $\mathbf{N} \in \{0,1\}^{w\times w}$ be an non-identity permutation matrix. $\Gamma^{(1)}$ differs in two matrices from $\Gamma^{(0)}$:

$$\Gamma^{(1)} = \left\{\mathbf{M}^{(1)}_{1,1} = \mathbf{N}, \mathbf{M}^{(1)}_{\ell+1,1} = \mathbf{N}^{-1}, \left\{\mathbf{M}^{(1)}_{i,b} = \mathbf{I}^{w\times w}\right\}_{i\in[1,h],b\in\{0,1\},(i,b)\notin\{(1,1),(\ell+1,1)\}}, \mathcal{P}_0, \mathcal{P}_1\right\}.$$

Both $\Gamma^{(0)}$, $\Gamma^{(1)}$ represent all-0 functionality.

The ρ^2 inputs are chosen as follows. For the prefixes $\left\{x^{(i)}\right\}_{i\in[\rho]}$, let $x^{(1)} = 1 \mid 0^{\nu-1}$. Let the rest be arbitrary from $0 \mid \{0,1\}^{\nu-1}$. The suffixes $\left\{y^{(j)}\right\}_{j\in[\rho]}$ are arbitrary from $\{0,1\}^{\ell-\nu}$. So that for $\Gamma^{(0)}$ the resulting $\mathbf{M}^{(0)}_{x^{(1)}} = \dots = \mathbf{M}^{(0)}_{x^{(\rho)}} = \mathbf{I}$; whereas for $\Gamma^{(1)}$, $\mathbf{M}^{(1)}_{x^{(1)}} = \mathbf{N}$, $\mathbf{M}^{(1)}_{x^{(2)}} = \dots = \mathbf{M}^{(1)}_{x^{(\rho)}} = \mathbf{I}$.

Here is a comparison of the resulting $\mathbf{B}$ matrices

$$\mathbf{B}^{(0)} = \begin{pmatrix} \alpha_{x^{(1)}}\mathbf{r}_J \mid \alpha'_{x^{(1)}}\mathbf{r}'_J \\ \alpha_{x^{(2)}}\mathbf{r}_J \mid \alpha'_{x^{(2)}}\mathbf{r}'_J \\ \cdots \\ \alpha_{x^{(\rho)}}\mathbf{r}_J \mid \alpha'_{x^{(\rho)}}\mathbf{r}'_J \end{pmatrix}, \quad \mathbf{B}^{(1)} = \begin{pmatrix} \alpha_{x^{(1)}}\mathbf{r}_J\mathbf{N} \mid \alpha'_{x^{(1)}}\mathbf{r}'_J \\ \alpha_{x^{(2)}}\mathbf{r}_J \mid \alpha'_{x^{(2)}}\mathbf{r}'_J \\ \cdots \\ \alpha_{x^{(\rho)}}\mathbf{r}_J \mid \alpha'_{x^{(\rho)}}\mathbf{r}'_J \end{pmatrix} \tag{42}$$

So $\mathsf{rank}(\mathbf{B}^{(0)}) = 2$, $\mathsf{rank}(\mathbf{B}^{(1)}) = 3$. Therefore w.h.p. (over the randomness in $\mathbf{D}_{x_2^{(1)}}$ ... $\mathbf{D}_{x_2^{(\rho)}}$) $\mathsf{rank}(\mathbf{X}_1^{(0)}) = \mathsf{rank}(\mathbf{C}^{(0)}) = 2 \cdot m^2$, $\mathsf{rank}(\mathbf{X}_1^{(1)}) = \mathsf{rank}(\mathbf{C}^{(1)}) = 2 \cdot m^2 + 1$. Hence $\Delta = 1$.

Justification of Claim 9.8.2. For the branching programs $\Gamma^{(0)}$ and $\Gamma^{(1)}$ we choose, the matrices controlled by input bits in $\mathcal{Y}$ are the same, so the properties of the resulting $\left\{\mathbf{Y}_k^{(0)}\right\}_{k\in[4]}$ are same to $\left\{\mathbf{Y}_k^{(1)}\right\}_{k\in[4]}$ (in other words the $\mathbf{Y}$ components do not cause the rank difference.) For the $\mathbf{X}_1$ and $\mathbf{X}_2$ components, following the similar analysis from $\mathbf{X}_3$, the ranks of $\mathbf{X}_1^{(1)}$ and $\mathbf{X}_2^{(1)}$ are larger or equal to the ranks of $\mathbf{X}_1^{(0)}$ and $\mathbf{X}_2^{(0)}$.

Our experiment shows that when $\rho \geq t^3 + t^2 \cdot m + t \cdot m^2 + m^3$, $\mathsf{rank}(\mathbf{V}^{(1)}) - \mathsf{rank}(\mathbf{V}^{(0)}) = 1$.

Extension to c-input-repeating BPs. For single input oblivious branching programs with c-input-repetitions (i.e. $h = c \cdot \ell$), the rank method still applies when $c > 2$. However, ρ (the dimension of $\mathbf{V}$) grows exponentially with c when converting the evaluation formula into 2-partition by recursively applying Formula (34). So the running time of the attack is $\mathrm{poly}(m,t)^c = \mathrm{poly}(\lambda,\ell)^c$ where λ is the security parameter, ℓ is the input length.

10 Witness Encryption Candidate

In this section, we present a witness encryption candidate for all of NP [GGSW13]. As mentioned in the introduction, our candidate relies on the observation from [GLW14] that to build witness encryption for general NP relations, it suffices to build witness encryption for CNF formulas, and that we can represent CNF formulas using read-once Type I branching programs.

10.1 Definition

We recall the definition of witness encryption from [GGSW13].

Definition 14 (Witness encryption [GGSW13])**.** A witness encryption scheme for an NP language L (with corresponding witness relation R) consists of the following two p.p.t. algorithms:

Encryption. $\mathsf{Enc}(1^\lambda, \Psi, \mu)$ takes as input a security parameter 1^λ, an instance $\Psi \in \{0,1\}^{\mathrm{poly}(\lambda)}$, and a message $\mu \in \{0,1\}$, outputs a ciphertext ct.

Decryption. $\mathsf{Dec}(ct, x)$ takes as input a ciphertext ct and string $x \in \{0,1\}^{\mathrm{poly}(\lambda)}$, outputs a message μ or the symbol $\perp$.

These algorithms satisfy

Correctness. For any security parameter λ, for any $\mu \in \{0,1\}$, and for any $\Psi \in L$ such that $R(\Psi, x)$ holds, we have that

$$\Pr[\mathsf{Dec}(\mathsf{Enc}(1^\lambda, \Psi, \mu), x) = \mu] \geq 1 - \mathrm{negl}(\lambda).$$

Soundness. For any p.p.t. adversary A, there exists a negligible function $\mathrm{negl}(\cdot)$ such that for any $\Psi \notin L$, we have

$$\left|\Pr[A(\mathsf{Enc}(1^\lambda, \Psi, 0)) = 1] - \Pr[A(\mathsf{Enc}(1^\lambda, \Psi, 1)) = 1]\right| \leq \mathrm{negl}(\lambda).$$

10.2 Construction

To build a witness encryption scheme for all of NP, it suffices to build one for the class of CNF formulas. We present a candidate for witness encryption based on the read-once Type I branching program for CNFs.

Construction 10.1 (Candidate witness encryption)**.** We construct a candidate witness encryption scheme for the class of CNF formula as follows:

Encryption. $\mathsf{Enc}(1^\lambda, \Psi, \mu)$ proceeds as follows:

- Apply Construction 6.4 on the CNF Ψ (of w clauses with h literals) to obtain a read-once matrix branching program $\{\mathbf{M}_{i,b} \in \{0,1\}^{w \times w}\}_{i \in [h], b \in \{0,1\}}$ for the CNF.
- Append μ to the bottom-right of $\{\mathbf{M}_{i,b}\}_{i \in [h], b \in \{0,1\}}$ to obtain

$$\left\{\mathbf{M}'_{i,b} = \begin{pmatrix} \mathbf{M}_{i,b} & \\ & \mu \end{pmatrix} \in \{0,1\}^{(w+1)\times(w+1)}\right\}_{i \in [h], b \in \{0,1\}}$$

  such that for all $\mathbf{x} \in \{0,1\}^h$,

$$\mathbf{M}'_{\mathbf{x}} = \begin{cases} \begin{pmatrix} \mathbf{0}^{w \times w} & \\ & \mu \end{pmatrix} & \text{if } \Psi(\mathbf{x}) = 1 \\ \begin{pmatrix} \neq \mathbf{0}^{w \times w} & \\ & \mu \end{pmatrix} & \text{if } \Psi(\mathbf{x}) = 0 \end{cases}.$$

- Set the parameters $n, m, q, B \in \mathbb{N}$, $\sigma \in \mathbb{R}^+$ s.t. $t = (w+1)n$, and the rest according to Remark 10.2. Sample $\left\{\mathbf{S}_{i,b} \leftarrow D_{\mathbb{Z},\sigma}^{n \times n}\right\}_{i \in [h], b \in \{0,1\}}$, $\mathbf{A}_h \leftarrow U(\mathbb{Z}_q^{t \times n})$ and compute

$$(\mathbf{A}_0, \{\mathbf{D}_{i,b}\}_{i \in [h], b \in \{0,1\}}) \leftarrow \mathsf{ggh.encode}(\gamma_\otimes, \left\{\mathbf{M}'_{i,b}\right\}_{i \in [h], b \in \{0,1\}}, \{\mathbf{S}_{i,b}\}_{i \in [h], b \in \{0,1\}}, \mathbf{A}_h, \sigma)$$

  and output

$$ct = \left(\mathbf{J}\mathbf{A}_0, \{\mathbf{D}_{i,b}\}_{i \in [h], b \in \{0,1\}}\right)$$

  where $\mathbf{J} = \mathbf{1}^{1 \times (w+1)} \otimes \mathbf{I}^{n \times n}$.

Decryption. $\mathsf{Dec}(ct, \mathbf{x})$ takes as input a ciphertext ct and string $\mathbf{x} \in \{0,1\}^h$.

- If ct cannot be parsed as $\left(\mathbf{J}\mathbf{A}_0, \{\mathbf{D}_{i,b}\}_{i\in[h],b\in\{0,1\}}\right)$ or $\Psi(\mathbf{x}) = 0$, outputs $\perp$.
- Else, outputs 0 if $\|\mathbf{J}\mathbf{A}_0 \cdot \mathbf{D}_{\mathbf{x}}\|_\infty \leq B$, and 1 otherwise.

Remark 10.2 (Parameters)**.** The parameters $n, m, q, B \in \mathbb{N}$, $\sigma \in \mathbb{R}^+$ are sampled under the following constraints for correctness and a hope of security (i.e. not for security reductions but matching the existing safety mechanisms). $m = \Omega(t \log q)$, $\sigma = \Omega(\sqrt{t \log q})$ for trapdoor functionality due to Lemma 3.10 and Lemma 3.11. $n = \Omega(\lambda \log q)$, $\chi = D_{\mathbb{Z},2\sqrt{\lambda}}$ due to Lemma 3.9 and Lemma 3.8. The noise threshold B is set according to Lemma 5.3

$$B \geq (w+1) \cdot h \cdot \left(m\sigma^2 \cdot \sqrt{n(w+1)}\right)^h \geq (w+1) \cdot h \cdot \left((w+1)\lambda \log^2 q\right)^{2.5h}.$$

Let $\epsilon \in (0,1)$. q is chosen s.t. $q \geq B \cdot \omega(\mathrm{poly}(\lambda))$ for correctness, $q \leq (\sigma/\lambda) \cdot 2^{\lambda^{1-\epsilon}}$ due to Lemma 3.8.

10.3 Relation to existing attacks

First, we note that using $\gamma_\otimes$-GGH15 encodings is necessary to prevent the subtraction attack in Section 9.3.1. All existing attack strategies on $\gamma_\otimes$-GGH15 encodings as used in our candidate require encodings of zeroes, which are not readily available in the witness encryption setting.

Remark 10.3 (candidate with γ_{diag})**.** A natural question is whether there is a candidate witness encryption scheme based on the γ_{diag}-GGH15 encodings. Namely, each $\hat{\mathbf{S}}_{i,b}$ possibly looks like $\begin{pmatrix} \mathbf{M}'_{i,b} & \\ & \mathbf{S}_{i,b} \end{pmatrix}$ where the CNF and message are potentially represented by $\left\{\mathbf{M}'_{i,b}\right\}_{i\in[h],b\in\{0,1\}}$. We do not know of such a candidate; naively substituting γ_{diag}-GGH encodings into the above candidate yields a witness encryption scheme that is susceptible to the subtraction attack.

11 Indistinguishability Obfuscation (iO) Candidate

In this section, we present an iO candidate for general Type I branching programs (cf. Def 6).

The basic idea is to start with our witness encryption candidate, that is, we apply the $\gamma_{\otimes\mathsf{diag}}$-GGH encodings to the matrix branching program representing our input circuit f. To prevent the advanced zeroizing + rank attack (cf. Section 9.3), we will pad the branching program with identity matrices so that it is $(\lambda+1)$-input-repeating. To prevent mixed-input attacks, we add "bundling matrices" whose subset product is $\mathbf{0}$ iff the inputs are well-formed.

11.1 Construction

We start with a Type I branching program $\Gamma = \{\mathbf{M}_{i,b}\}_{i\in[h],b\in\{0,1\}}$. We assume that it reads the bits in order $1, 2, \ldots, \ell, 1, 2, \ldots, \ell$ and so on. WLOG, we may assume that Γ is $(\lambda+1)$-input-repeating by padding with identity matrices on the right.

Construction 11.1 (candidate iO)**.** We construct a candidate iO scheme for the class of $(\lambda+1)$-repeating Type I branching programs as follows:

Obfuscation On input $\{\mathbf{M}_{i,b} \in \{0,1\}^{w\times w}\}_{i\in[h],b\in\{0,1\}}$ where $h = (\lambda+1)\ell$,

- Sample bundling matrices $\{\mathbf{R}_{i,b}\}_{i\in[h],b\in\{0,1\}}$ as follows. Each $\mathbf{R}_{i,b} \in \mathbb{Z}^{2n\ell\times 2n\ell}$ will be of the form $\mathrm{diag}(\mathbf{R}_{i,b}^{(1)},\ldots,\mathbf{R}_{i,b}^{(\ell)})$ where

$$\mathbf{R}_{i,b}^{(k)} = \begin{cases} \mathbf{I}^{2n\times 2n} & \text{if } \iota(i) \neq k \\ \begin{pmatrix} \tilde{\mathbf{R}}_{i,b}^{(k)} & \\ & \mathbf{I}^{n\times n} \end{pmatrix}, \tilde{\mathbf{R}}_{i,b}^{(k)} \leftarrow D_{\mathbb{Z},\sigma}^{n\times n} & \text{if } \iota(i) = k \text{ and } i \leq \lambda\ell \\ \begin{pmatrix} -\mathbf{I}^{n\times n} & \\ & \tilde{\mathbf{R}}_{k,b}^{(k)} \cdot \tilde{\mathbf{R}}_{k+\ell,b}^{(k)} \cdot \cdots \cdot \tilde{\mathbf{R}}_{k+(\lambda-1)\ell,b}^{(k)} \end{pmatrix} & \text{if } \iota(i) = k \text{ and } i > \lambda\ell \end{cases}$$

  In particular, for all $\mathbf{x}' \in \{0,1\}^h$,

$$(\mathbf{1}^{1\times 2\ell} \otimes \mathbf{I}^{n\times n}) \cdot \mathbf{R}_{\mathbf{x}'} \cdot (\mathbf{1}^{2\ell\times 1} \otimes \mathbf{I}^{n\times n}) = \mathbf{0}^{n\times n} \iff \mathbf{x}' \in \varpi(\{0,1\}^\ell)$$

- Set the parameters $n, m, q, B \in \mathbb{N}$, $\sigma \in \mathbb{R}^+$ s.t. $t = (w+2n\ell)\cdot n$, and the rest according to Remark 11.2. Sample $\left\{\mathbf{S}_{i,b} \leftarrow D_{\mathbb{Z},\sigma}^{n\times n}\right\}_{i\in[h],b\in\{0,1\}}$ and compute

$$\begin{aligned} \mathbf{J} &:= \mathbf{1}^{1\times(w+2n\ell)} \otimes \mathbf{I}^{n\times n} \\ \hat{\mathbf{S}}_{i,b} &:= \begin{pmatrix} \mathbf{M}_{i,b} \otimes \mathbf{S}_{i,b} & \\ & \mathbf{R}_{i,b} \otimes \mathbf{S}_{i,b} \end{pmatrix} \\ \mathbf{L} &:= (\mathbf{1}^{(w+2n\ell)\times 1} \otimes \mathbf{I}^{n\times n}) \end{aligned}$$

  In particular, for all $\mathbf{x}' \in \{0,1\}^h$,

$$\mathbf{J} \cdot \hat{\mathbf{S}}_{\mathbf{x}'} \cdot \mathbf{L} = \begin{cases} \mathbf{0} & \text{if } f(\mathbf{x}) = 1, \mathbf{x}' = \varpi(\mathbf{x}) \\ \neq \mathbf{0} & \text{if } f(\mathbf{x}) = 0, \mathbf{x}' = \varpi(\mathbf{x}) \text{ since } \mathbf{M}_{\mathbf{x}'} \neq \mathbf{0} \\ \neq \mathbf{0} & \text{if } \mathbf{x}' \notin \varpi(\{0,1\}^\ell) \text{ due to bundling matrices} \end{cases}$$

- Sample $\mathbf{A}_h \leftarrow U(\mathbb{Z}_q^{n\times n})$, all the error terms $\mathbf{E}_{i,b}$ from $D_{\mathbb{Z},\sigma}$ with the corresponding dimensions. Compute and output

$$\begin{aligned} \mathbf{A}_J &:= \mathbf{J}\mathbf{A}_0 \in \mathbb{Z}^{n\times m} \\ \mathbf{D}_{i,b} &\leftarrow \mathbf{A}_{i-1}^{-1}(\hat{\mathbf{S}}_{i,b} \cdot \mathbf{A}_i + \mathbf{E}_{i,b}) \in \mathbb{Z}^{m\times m}, i = 1,\ldots,h-1 \\ \mathbf{D}_{h,b} &\leftarrow \mathbf{A}_{h-1}^{-1}(\hat{\mathbf{S}}_{h,b} \cdot \mathbf{L} \cdot \mathbf{A}_h + \mathbf{E}_{h,b}) \in \mathbb{Z}^{m\times n} \end{aligned}$$

Evaluation Output 1 if $\|\mathbf{A}_J \cdot \mathbf{D}_{\varpi(\mathbf{x})}\|_\infty \leq B$, and 0 otherwise.

Remark 11.2 (Parameters)**.** The parameters $n, m, q, B \in \mathbb{N}$, $\sigma \in \mathbb{R}^+$ are sampled under the following constraints for correctness and a hope of security (i.e. not for security reductions but matching the existing safety mechanisms). $m = \Omega(t\log q)$, $\sigma = \Omega(\sqrt{t\log q})$ for trapdoor functionality due to Lemma 3.10 and Lemma 3.11. $n = \Omega(\lambda \log q)$, $\chi = D_{\mathbb{Z},2\sqrt{\lambda}}$ due to Lemma 3.9 and Lemma 3.8. The noise threshold B is set according to Lemma 5.3

$$B \geq (w+2n\ell)\cdot h \cdot \left(m\sigma^2 \cdot \sqrt{n(w+2n\ell)\sigma}\right)^h.$$

Let $\epsilon \in (0,1)$. q is chosen s.t. $q \geq B \cdot \omega(\mathrm{poly}(\lambda))$ for correctness, $q \leq (\sigma/\lambda)\cdot 2^{\lambda^{1-\epsilon}}$ due to Lemma 3.8.

11.2 Discussion

As with the witness encryption candidate, using $\gamma_\otimes$ is necessary to prevent the subtraction attack; in particular, the bundling matrices $\{\mathbf{R}_{i,b}\}$ themselves do not prevent the subtraction attack.

Next, we need input-repetition to prevent the rank attacks in Section 9.3.2.

Finally, we need the bundling matrices to prevent mixed-input attacks:

- A difference from previous candidates is that we embed the bundling matrices into the diagonal (as opposed to a product or a tensor with the $\mathbf{M}_{i,b}$'s); another is that the product of the bundling matrices is $\mathbf{0}$ instead of the identity.
- Note that the bundling matrices $\{\mathbf{R}_{i,b}\}$ also interacts with $\{\mathbf{S}_{i,b}\}$ to prevent subtraction attacks on the bundling matrices.

Remark 11.3 (variants with additional randomization)**.** For both the witness and iO candidate, we can append a random small $\mathbf{S}'_{i,b} \in \mathbb{Z}^{n\times n}$ to the bottom right of $\hat{\mathbf{S}}_{i,b}$. For functionality, in the iO candidate we will need to modify the right-bookend $\mathbf{L}$ to be $(\mathbf{1}^{(w+2n\ell)\times 1} \mid 0)^T \otimes \mathbf{I}^{n\times n}$; in the witness encryption candidate let $\mathbf{A}_h \leftarrow \begin{pmatrix} U(\mathbb{Z}_q^{t\times n}) \\ \mathbf{0}^{n\times n} \end{pmatrix}$.

11.3 Sanity check

We make a sanity check to ensure that the candidate is not trivially broken by making illegal evaluations (i.e. by the subtraction attack or the mixed-input attack).

Observe that for all $\mathbf{x}' \in \{0,1\}^h$, with high probability over the randomness of the encoding,

$$\mathbf{J}\cdot\hat{\mathbf{S}}_{\mathbf{x}'}\cdot\mathbf{L} = \begin{cases} 0\cdot\mathbf{S}_{\mathbf{x}'} & \text{if } f(\mathbf{x})=1, \mathbf{x}'=\varpi(\mathbf{x}) \\ (\neq 0)\cdot\mathbf{S}_{\mathbf{x}'} & \text{if } f(\mathbf{x})=0, \mathbf{x}'=\varpi(\mathbf{x}) \text{ since } \mathbf{M}_{\mathbf{x}'}\neq\mathbf{0} \\ (\neq 0)\cdot\mathbf{S}_{\mathbf{x}'} & \text{if } \mathbf{x}'\notin\varpi(\{0,1\}^\ell) \text{ due to bundling matrices} \end{cases}$$

Here, the $\neq 0$ constants above have low-norm and depend only on $\mathbf{x}'$ and either Γ or the bundling matrices (and not the $\mathbf{S}_{i,b}$'s).

Now, consider an extremely restricted adversary that instead of seeing $\mathbf{A}_J, \{\mathbf{D}_{i,b}\}_{i\in[h],b\in\{0,1\}}$, only gets oracle access to

$$\mathbf{x}' \mapsto \lfloor \mathbf{A}_J\mathbf{D}_{\mathbf{x}'}\rceil_p .$$

Note that such an adversary can already implement the subtraction attack and certain mixed-input attacks but not the rank attack and other zeroizing attacks. Observe that

$$\lfloor \mathbf{A}_J\mathbf{D}_{\mathbf{x}'}\rceil_p = \begin{cases} \mathbf{0}^{n\times n} & \text{if } f(\mathbf{x})=1, \mathbf{x}'=\varpi(\mathbf{x}) \\ \lfloor(\neq 0)\cdot\mathbf{S}_{\mathbf{x}'}\mathbf{A}_h\rceil_p & \text{if } f(\mathbf{x})=0, \mathbf{x}'=\varpi(\mathbf{x}) \text{ since } \mathbf{M}_{\mathbf{x}'}\neq\mathbf{0} \\ \lfloor(\neq 0)\cdot\mathbf{S}_{\mathbf{x}'}\mathbf{A}_h\rceil_p & \text{if } \mathbf{x}'\notin\varpi(\{0,1\}^\ell) \text{ due to bundling matrices} \end{cases}$$

By the security of the BLMR PRF with $\mathbf{A}_h$ as the seed, we can computationally simulate the output of the oracle on input $\mathbf{x}'$ (given oracle access to f) as follows:

- $\mathbf{0}^{n\times n}$ if $f(\mathbf{x})=1, \mathbf{x}'=\varpi(\mathbf{x})$

- $U(\mathbb{Z}_p^{n\times n})$ otherwise

Of course, assuming the adversary only gets the rounded output is unrealistic. Indeed, security against such adversaries provides no indication whatsoever about the security of the iO candidate. Nonetheless, proving security against these extremely restricted adversaries constitutes a useful sanity check against the subtraction attacks and variants there-of. As mentioned in Section 1.4, we believe that an important open problem is to understand security against adversaries that get the output without rounding.

References

[ADGM17] Daniel Apon, Nico Döttling, Sanjam Garg, and Pratyay Mukherjee. Cryptanalysis of indistinguishability obfuscations of circuits over GGH13. In *ICALP*, volume 80 of *LIPIcs*, pages 38:1–38:16. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017.

[Ajt99] Miklós Ajtai. Generating hard instances of the short basis problem. In Jirí Wiedermann, Peter van Emde Boas, and Mogens Nielsen, editors, *Automata, Languages and Programming, 26th International Colloquium, ICALP'99, Prague, Czech Republic, July 11-15, 1999, Proceedings*, volume 1644 of *LNCS*, pages 1–9. Springer, 1999.

[AP11] Joël Alwen and Chris Peikert. Generating shorter bases for hard random lattices. *Theory of Computing Systems*, 48(3):535–553, 2011.

[Bar86] David A. Mix Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in nc^1. In Juris Hartmanis, editor, *STOC*, pages 1–5. ACM, 1986.

[BF14] Jean-François Biasse and Claus Fieker. Subexponential class group and unit group computation in large degree number fields. *LMS Journal of Computation and Mathematics*, 17(A):385–403, 2014.

[BGK+14] Boaz Barak, Sanjam Garg, Yael Tauman Kalai, Omer Paneth, and Amit Sahai. Protecting obfuscation against algebraic attacks. In *EUROCRYPT*, volume 8441 of *LNCS*, pages 221–238. Springer, 2014.

[BKM17] Dan Boneh, Sam Kim, and Hart William Montgomery. Private puncturable prfs from standard lattice assumptions. In *EUROCRYPT (1)*, volume 10210 of *Lecture Notes in Computer Science*, pages 415–445, 2017.

[BLMR13] Dan Boneh, Kevin Lewi, Hart William Montgomery, and Ananth Raghunathan. Key homomorphic prfs and their applications. In *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, pages 410–428, 2013.

[BLP+13] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. Classical hardness of learning with errors. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 575–584. ACM, 2013.

[BLW17] Dan Boneh, Kevin Lewi, and David J. Wu. Constraining pseudorandom functions privately. In *Public-Key Cryptography - PKC 2017 - 20th IACR International Conference on Practice and Theory in Public-Key Cryptography, Amsterdam, The Netherlands, March 28-31, 2017, Proceedings, Part II*, pages 494–524, 2017.

[BPR12] Abhishek Banerjee, Chris Peikert, and Alon Rosen. Pseudorandom functions and lattices. In *EUROCRYPT*, pages 719–737, 2012.

[BR13] Zvika Brakerski and Guy N. Rothblum. Obfuscating conjunctions. In *CRYPTO, Part II*, pages 416–434, 2013.

[BS03] Dan Boneh and Alice Silverberg. Applications of multilinear forms to cryptography. *Contemporary Mathematics*, 324(1):71–90, 2003.

[BS16] Jean-François Biasse and Fang Song. Efficient quantum algorithms for computing class groups and solving the principal ideal problem in arbitrary degree number fields. In *SODA*, pages 893–902. SIAM, 2016.

[BTVW17] Zvika Brakerski, Rotem Tsabary, Vinod Vaikuntanathan, and Hoeteck Wee. Private constrained prfs (and more) from LWE. In *Theory of Cryptography - 15th International Conference, TCC 2017, Baltimore, MD, USA, November 12-15, 2017, Proceedings, Part I*, pages 264–302, 2017.

[BVWW16] Zvika Brakerski, Vinod Vaikuntanathan, Hoeteck Wee, and Daniel Wichs. Obfuscating conjunctions under entropic ring LWE. In *ITCS*, pages 147–156. ACM, 2016.

[CC17] Ran Canetti and Yilei Chen. Constraint-hiding constrained PRFs for NC^1 from LWE. In *EUROCRYPT 2017, Part I*, pages 446–476, 2017.

[CDPR16] Ronald Cramer, Léo Ducas, Chris Peikert, and Oded Regev. Recovering short generators of principal ideals in cyclotomic rings. In *EUROCRYPT (2)*, volume 9666 of *LNCS*, pages 559–585. Springer, 2016.

[CGH+15] Jean-Sébastien Coron, Craig Gentry, Shai Halevi, Tancrede Lepoint, Hemanta K Maji, Eric Miles, Mariana Raykova, Amit Sahai, and Mehdi Tibouchi. Zeroizing without low-level zeroes: New mmap attacks and their limitations. In *Advances in Cryptology–CRYPTO 2015*, pages 247–266. Springer, 2015.

[CGH17] Yilei Chen, Craig Gentry, and Shai Halevi. Cryptanalyses of candidate branching program obfuscators. In *EUROCRYPT (3)*, volume 10212 of *Lecture Notes in Computer Science*, pages 278–307, 2017.

[CHKP12] David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. *Journal of cryptology*, 25(4):601–639, 2012.

[CHL+15] Jung Hee Cheon, Kyoohyung Han, Changmin Lee, Hansol Ryu, and Damien Stehlé. Cryptanalysis of the multilinear map over the integers. In *EUROCRYPT (1)*, volume 9056 of *LNCS*, pages 3–12. Springer, 2015.

[CLLT16] Jean-Sébastien Coron, Moon Sung Lee, Tancrède Lepoint, and Mehdi Tibouchi. Cryptanalysis of GGH15 multilinear maps. In *CRYPTO (2)*, volume 9815 of *LNCS*, pages 607–628. Springer, 2016.

[CLLT17] Jean-Sébastien Coron, Moon Sung Lee, Tancrède Lepoint, and Mehdi Tibouchi. Zeroizing attacks on indistinguishability obfuscation over CLT13. In *Public Key Cryptography (1)*, volume 10174 of *Lecture Notes in Computer Science*, pages 41–58. Springer, 2017.

[CLT13] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. Practical multilinear maps over the integers. In *CRYPTO (1)*, pages 476–493, 2013.

[DGK+10] Yevgeniy Dodis, Shafi Goldwasser, Yael Tauman Kalai, Chris Peikert, and Vinod Vaikuntanathan. Public-key encryption schemes with auxiliary inputs. In *Theory of Cryptography, 7th Theory of Cryptography Conference, TCC 2010, Zurich, Switzerland, February 9-11, 2010. Proceedings*, pages 361–381, 2010.

[FRS17] Rex Fernando, Peter M. R. Rasmussen, and Amit Sahai. Preventing CLT attacks on obfuscation with linear overhead. In *ASIACRYPT (3)*, volume 10626 of *Lecture Notes in Computer Science*, pages 242–271. Springer, 2017.

[GGH13a] Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In *EUROCRYPT*, pages 1–17, 2013.

[GGH+13b] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *FOCS*, pages 40–49, 2013.

[GGH15] Craig Gentry, Sergey Gorbunov, and Shai Halevi. Graph-induced multilinear maps from lattices. In *TCC 2015, Part II*, pages 498–527, 2015.

[GGSW13] Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. Witness encryption and its applications. In *STOC*, pages 467–476, 2013.

[GKW17a] Rishab Goyal, Venkata Koppula, and Brent Waters. Lockable obfuscation. In *FOCS*, pages 612–621, 2017.

[GKW17b] Rishab Goyal, Venkata Koppula, and Brent Waters. Separating semantic and circular security for symmetric-key bit encryption from the learning with errors assumption. In *EUROCRYPT (2)*, pages 528–557, 2017.

[GKW18] Rishab Goyal, Venkata Koppula, and Brent Waters. Collusion resistant traitor tracing from learning with errors. In *STOC*, 2018.

[GLW14] Craig Gentry, Allison B. Lewko, and Brent Waters. Witness encryption from instance independent assumptions. In *CRYPTO (1)*, volume 8616 of *Lecture Notes in Computer Science*, pages 426–443. Springer, 2014.

[GMM+16] Sanjam Garg, Eric Miles, Pratyay Mukherjee, Amit Sahai, Akshayaram Srinivasan, and Mark Zhandry. Secure obfuscation in a weak multilinear map model. In *TCC (B2)*, volume 9986 of *Lecture Notes in Computer Science*, pages 241–268, 2016.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC*, pages 197–206, 2008.

[Hal15] Shai Halevi. Graded encoding, variations on a scheme. Cryptology ePrint Archive, Report 2015/866, 2015.

[HHSS17] Shai Halevi, Tzipora Halevi, Victor Shoup, and Noah Stephens-Davidowitz. Implementing BP-obfuscation using graph-induced encoding. In *ACM CCS*, pages 783–798, 2017.

[Kle00] Philip N. Klein. Finding the closest lattice vector when it's unusually close. In *SODA*, pages 937–941. ACM/SIAM, 2000.

[KW16] Venkata Koppula and Brent Waters. Circular security separations for arbitrary length cycles from LWE. In *CRYPTO (2)*, pages 681–700, 2016.

[MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *Advances in Cryptology–EUROCRYPT 2012*, pages 700–718. Springer, 2012.

[MR07] Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on Gaussian measure. *SIAM Journal on Computing*, 37(1):267–302, 2007.

[MSZ16] Eric Miles, Amit Sahai, and Mark Zhandry. Annihilation attacks for multilinear maps: Cryptanalysis of indistinguishability obfuscation over GGH13. In *CRYPTO (2)*, volume 9815 of *LNCS*, pages 629–658. Springer, 2016.

[Pei09] Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In *STOC*, pages 333–342, 2009.

[PR06] Chris Peikert and Alon Rosen. Efficient collision-resistant hashing from worst-case assumptions on cyclic lattices. In *Theory of Cryptography*, pages 145–166. Springer, 2006.

[PRS17] Chris Peikert, Oded Regev, and Noah Stephens-Davidowitz. Pseudorandomness of ring-lwe for any ring and modulus. In *STOC*, pages 461–473. ACM, 2017.

[PS18] Chris Peikert and Sina Shiehian. Privately constraining and programming PRFs, the LWE way. In *PKC*, 2018.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005*, pages 84–93. ACM, 2005.

[Reg09] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 56(6), 2009.

[WZ17] Daniel Wichs and Giorgos Zirdelis. Obfuscating compute-and-compare programs under LWE. In *FOCS*, pages 600–611, 2017.

A Attacking the iO candidates based on GGH13

The similar analysis techniques mentioned in Section 9 can also be used to extend the previous attacks on candidate branching program obfuscators based on GGH13.

A.1 A distinguishing attack for iO candidates using GGH13

We demonstrate a distinguishing attack for the iO candidates built from the construction in Section 9.1 using GGH13 graded encoding scheme [GGH13a]. As mentioned in the summary in Section 9.2, our attack improves upon [CGH17] in terms of the type of BPs (remove the input-partition requirement), upon [ADGM17] in terms of the ability to handle the diagonal paddings.

Conceptually, the attack also use the some identities of the matrices collected from evaluations of zeros, but this time (i.e. for GGH13) we use the eigenvalues instead of the rank as the distinguishing factor.

A.1.1 Brief recap of GGH13 in the context of branching program obfuscation

Let $L = \mathbb{Z}[X]/(\phi_m(X))$ where ϕ_m is the m^{th} cyclotomic polynomial. $K = \mathbb{Q}[X]/(\phi_m(X))$ be the fractional field of L. The core secret parameter in the GGH13 encoding scheme is a small $g \in L$ (sampled from small Gaussian distribution), such that the inverse $g^{-1} \in K$ is also small. Let $\mathcal{I} = \langle g \rangle = g \cdot L$ be the ideal generated by g in L. The plaintext space of the GGH13 is the quotient ring $R = L/\mathcal{I}$, and we typically choose g so that this plaintext space is isomorphic to some prime field $\mathbb{F}_p$. Other parameters of the scheme are an integer modulus $q \gg p$ and the multi-linearity degree κ (which are public), and a random secret denominator $z \in L/qL$ (which is kept secret). Plaintext elements are encoded relative to levels between 0 and κ.

The encoding of $s \in R$ at level 0 is a short representative of the coset of the ideal shifted by s, i.e. $c \in s + \mathcal{I}$, $\|c\| \leq q^{1/(2\kappa+1)}$. To encode at level i, compute $c/z^i \pmod q$. The zero-test parameter is $p_{zt} = \eta \cdot z^\kappa / g$, with $\|\eta\| \leq q^{1/2}$. Additions and multiplications are simply adding and multiplying the encodings in L/qL, with the restrictions that correctness only holds when adding on the same level, or multiplying below the maximum level κ. To zero-test, multiply the (potential) top-level encoding c/z^κ by p_{zt} (modulo q). If c encodes zero then $c \in \mathcal{I}$, hence $c = c' \cdot g$, and therefore $c \cdot p_{zt} = \eta c'$, which is small since both η and c' are much smaller than q.

When GGH13 is used in branching program obfuscation (cf. Section 9.1), we set $\kappa = h$. Each matrix $\hat{\mathbf{S}}_{i,b} \in R^{t \times t}$ from Eqn. (20) is encoded (entry-wise) on level 1. The bookends $\mathbf{J}, \mathbf{L}$ are merged into the first and last matrices. p_{zt} is merged in the first matrices. Denote the resulting matrices as

$$\begin{aligned}
&\left\{\mathbf{C}_{1,b} := p_{zt} \cdot \left(\mathbf{J} \times \hat{\mathbf{S}}_{1,b} + g \cdot \mathbf{E}_{1,b}\right) / z \in L_q^{1\times t}\right\}_{b\in\{0,1\}} \\
&\left\{\mathbf{C}_{i,b} := \left(\hat{\mathbf{S}}_{i,b} + g \cdot \mathbf{E}_{i,b}\right) / z \in L_q^{t\times t}\right\}_{i\in[2,h-1],b\in\{0,1\}} \\
&\left\{\mathbf{C}_{h,b} := \left(\hat{\mathbf{S}}_{h,b}\mathbf{L} + g \cdot \mathbf{E}_{h,b}\right) / z \in L_q^{t\times 1}\right\}_{b\in\{0,1\}}.
\end{aligned} \tag{43}$$

The obfuscated code consists of $\{\mathbf{C}_{i,b}\}_{i\in[1,h],b\in\{0,1\}}$. To evaluate on $x \in \{0,1\}^\ell$, compute $v(x) = \mathbf{C}_{\varpi(x)} \in L_q$, output 0 if $\|v(x)\| \leq q^{1-1/(4h+3)}$, 1 otherwise.

A.1.2 The attack algorithm and analysis

The attack algorithm follows the steps from [CGH17] in the 3-input-partition setting, and use the formulas from [ADGM17, CLLT17] to convert the input-repeating programs into input-partitioned.

The attack from [CGH17] runs the following 3 steps:

1. Compute the determinant of some matrix to recover a basis of the ideal $\mathcal{I}$.
2. Extract the ratio of bundling scalars from the eigenvalue of some matrix mod $\mathcal{I}$.
3. Use the bundling scalars to conduct a simplified annihilation attack [MSZ16].

If you are willing to spend classical subexponential time or use a quantum computer, then from a basis of $\mathcal{I}$ obtained from Step 1, you can find a short generator of $\mathcal{I}$ [BF14, BS16, CDPR16] and break GGH13 completely, instead of continuing with Steps 2 and 3.

In the input-repeating setting, we can finish a classical polynomial time distinguishing attack in 2 steps. The simplification comes from the fact that there exists functionally equivalent input-repeating permutation BPs with features that are detectable in 2 steps; whereas for input-partition permutation BPs, such a distinguishing feature does not exist due to [CGH17, Lemma 2.2].

Similar to the situation in GGH15, our analysis uses heuristic assumptions on the the linear independence of matrices resulting from the products of Gaussians.

Attack for 2-input-repeating branching programs. We start from 2-input-repeating branching programs. Assume $h = 2\ell$, the input bit sequence is "$1, 2, ..., \ell, 1, 2, ..., \ell$", i.e. $\varpi(u) = u \mid u$. Let $\rho = t^3$ (recall from Eqn. (20) that $t = 2(w + n)$).

In the attack we need to partition the ℓ input bits into 3 intervals for several times (note that the boundaries of the partitions are not necessarily fixed across different steps in the attack; we will clarify the restrictions on the boundaries every time.) Denote the partition as $[\ell] = \mathcal{X} \mid \mathcal{Y} \mid \mathcal{Z}$; let the corresponding BP steps be partitioned into 6 intervals $[h] = \mathcal{X}_1 \mid \mathcal{Y}_1 \mid \mathcal{Z}_1 \mid \mathcal{X}_2 \mid \mathcal{Y}_2 \mid \mathcal{Z}_2$ where for $\mathcal{W} \in \{\mathcal{X}, \mathcal{Y}, \mathcal{Z}\}$, the steps in $\mathcal{W}_1, \mathcal{W}_2$ are controlled by the input bits in $\mathcal{W}$. Denote $\varpi(u^{(i,j,k)}) = \varpi(x^{(i)} \mid y^{(j)} \mid z^{(k)}) =: x_1^{(i)} \mid y_1^{(j)} \mid z_1^{(k)} \mid x_2^{(i)} \mid y_2^{(j)} \mid z_2^{(k)}$.

A subroutine in the attack. The real attack uses following subroutine. The similar subroutine appeared in [CHL+15, CGH+15, CGH17, CLLT17] in the context of attacking GGH13 or CLT13.

Algorithm A.1 (A subroutine). Given as inputs the obfuscated code $\{\mathbf{C}_{i,b}\}_{i\in[1,h],b\in\{0,1\}}$, ρ prefixes $\{x^{(i)}\}_{i\in[\rho]}$, a single string $y^{(*)}$, and ρ suffixes $\{z^{(k)}\}_{k\in[\rho]}$. The subroutine evaluates the obfuscated code on ρ^2 inputs of the form $\{u^{(i,*,k)}\}_{i\in[\rho],k\in[\rho]}$. Output a matrix $\mathbf{V}^{(*)} \in L_q^{\rho\times\rho}$ whose $(i\times k)^{th}$ entry is $v^{(i,*,k)} := \mathbf{C}_{\varpi(u^{(i,*,k)})}$.

Analysis of the subroutine. We open up each $v^{(i,*,k)}$ obtained from Alg. A.1.

$$\begin{aligned} v^{(i,*,k)} &= \mathbf{C}_{x_1^{(i)}} \cdot \mathbf{C}_{y_1^{(*)}} \cdot \mathbf{C}_{z_1^{(k)}} \cdot \mathbf{C}_{x_2^{(i)}} \cdot \mathbf{C}_{y_2^{(*)}} \cdot \mathbf{C}_{z_2^{(k)}} \\ &= \left(\mathsf{vec}(\mathbf{C}_{x_2^{(i)}})^T \otimes \mathbf{C}_{x_1^{(i)}}\right) \cdot \left(\mathbf{C}_{y_2^{(*)}} \otimes \mathbf{I}^{t\times t} \otimes \mathbf{C}_{y_1^{(*)}}\right) \cdot \mathsf{vec}\left(\mathbf{C}_{z_2^{(k)}}{}^T \otimes \mathbf{C}_{z_1^{(k)}}\right) \end{aligned} \tag{44}$$

where the subscripts slightly abuse the subset product notation, similar to the notation convention used in Eqn. (24) (i.e. $\mathbf{C}_{y_1^{(*)}} = \prod_{i=1}^{|\mathcal{Y}_1|} \mathbf{C}_{\nu+i,y_i^{(*)}}$, etc.) The second equality follows Formula (34).

If $\forall i \in [\rho], k \in [\rho]$, $u^{(i,*,k)}$ is the zero of the function, then Eqn. (44) holds over L (not only over L_q). Therefore we have

$$\begin{aligned}\mathbf{V}^{(*)} &:= \begin{pmatrix} v^{(1,*,1)} & \dots & v^{(1,*,\rho)} \\ \dots & \dots & \dots \\ v^{(\rho,*,1)} & \dots & v^{(\rho,*,\rho)} \end{pmatrix} \\ &= \underbrace{\begin{pmatrix} \dots \\ \mathsf{vec}(\mathbf{C}_{x_2^{(i)}})^T \otimes \mathbf{C}_{x_1^{(i)}} \\ \dots \end{pmatrix}}_{:=\mathbf{X}\in L^{\rho\times\rho}} \cdot \underbrace{\left(\mathbf{C}_{y_2^{(*)}} \otimes \mathbf{I}^{t\times t} \otimes \mathbf{C}_{y_1^{(*)}}\right)}_{:=\mathbf{Y}^{(*)}\in L^{\rho\times\rho}} \cdot \underbrace{\left(\dots \middle| \mathsf{vec}(\mathbf{C}_{z_2^{(k)}})^T \otimes \mathbf{C}_{z_1^{(k)}} \middle| \dots\right)}_{:=\mathbf{Z}\in L^{\rho\times\rho}} \in L^{\rho\times\rho}\end{aligned} \tag{45}$$

Claim A.2. $\mathbf{X}$, $\mathbf{Y}^{(*)}$ and $\mathbf{Z}$ in Eqn. (45) are non-singular over L heuristically.

First we check the singularity of $\mathbf{Y}^{(*)} = \mathbf{C}_{y_2^{(*)}} \otimes \mathbf{I}^{t\times t} \otimes \mathbf{C}_{y_1^{(*)}}$, which is a tensor product of 3 square matrices. If both $\mathbf{C}_{y_2^{(*)}}$ and $\mathbf{C}_{y_1^{(*)}}$ are non-singular, $\mathbf{Y}^{(*)}$ is non-singular. So we take a closer look at $\mathbf{C}_{y_1^{(*)}}$:

$$\mathbf{C}_{y_1^{(*)}} = \prod_{i=1}^{|\mathcal{Y}_1|} \mathbf{C}_{\nu+i,y_i^{(*)}} = \prod_{i=1}^{|\mathcal{Y}_1|} \left(\hat{\mathbf{S}}_{\nu+i,y_i^{(*)}} + g \cdot \mathbf{E}_{\nu+i,y_i^{(*)}}\right) \tag{46}$$

We claim that each $\mathbf{E}_{\nu+i,y_i^{(*)}} \in L^{t\times t}$ matrix is non-singular whp, since each entry of the matrix is sampled independently from the Gaussian coset of a fixed value with sufficiently large width; the independence holds across these $\mathbf{E}$ matrices. So the product is still non-singular whp.

The observation from Eqn. (46) in fact holds for all $\mathbf{C}_{y_2^{(*)}}$, $\mathbf{C}_{x_1^{(i)}}$, $\mathbf{C}_{z_1^{(k)}}$, $\mathbf{C}_{x_2^{(i)}}$, $\mathbf{C}_{z_2^{(k)}}$, due to the $g \cdot \mathbf{E}$ components. In other words in analyzing the rank of these matrices over L, we do not need to worry about the $\hat{\mathbf{S}}$ components. This immediately justify the non-singularity of $\mathbf{Y}^{(*)}$.

For $\mathbf{X}$, each row is a tensor product of 2 random'ish row vectors where the randomness come from the $g \cdot \mathbf{E}$ components; then we use the heuristics about linear independence across rows and columns similar to the analysis for GGH15. The same methodology applies to $\mathbf{Z}$.

The running example. To assist the description of the rest of the analysis, we give the running example of $\Gamma^{(0)}$ and $\Gamma^{(1)}$ first.

Example A.1. For $r \in [\ell]$ such that $\log(\rho) < r < \ell - \log(\rho)$. Let $\mathbf{N}$ be a non-identity permutation matrix. Here are two BPs that compute the 0-function:

$\Gamma^{(0)}$	0 :	$\mathbf{I} \dots \mathbf{I}$	$\mathbf{I}$	$\mathbf{I} \dots \mathbf{I}$	$\mathbf{I} \dots \mathbf{I}$	$\mathbf{I}$	$\mathbf{I} \dots \mathbf{I}$
	1 :	$\mathbf{I} \dots \mathbf{I}$	$\mathbf{I}$	$\mathbf{I} \dots \mathbf{I}$	$\mathbf{I} \dots \mathbf{I}$	$\mathbf{I}$	$\mathbf{I} \dots \mathbf{I}$
$\Gamma^{(1)}$	0 :	$\mathbf{I} \dots \mathbf{I}$	$\mathbf{I}$	$\mathbf{I} \dots \mathbf{I}$	$\mathbf{I} \dots \mathbf{I}$	$\mathbf{I}$	$\mathbf{I} \dots \mathbf{I}$
	1 :	$\mathbf{I} \dots \mathbf{I}$	$\mathbf{N}$	$\mathbf{I} \dots \mathbf{I}$	$\mathbf{I} \dots \mathbf{I}$	$\mathbf{N}^{-1}$	$\mathbf{I} \dots \mathbf{I}$
BP Steps :		$1 \dots r-1$	r	$r+1 \dots \ell$	$\ell+1 \dots \ell+r-1$	$\ell+r$	$\ell+r+1 \dots 2\ell$
input bits :		$1 \dots r-1$	r	$r+1 \dots \ell$	$1 \dots r-1$	r	$r+1 \dots \ell$

(47)

Looking ahead, when the input bits are partition as $[\ell] = \mathcal{X} \mid \mathcal{Y} \mid \mathcal{Z}$ in the attack steps, the boundaries of the partitions are different across steps. In the first step, the r^{th} bit does *not* live in the $\mathcal{Y}$ interval; in the second step, the r^{th} bit does live in the $\mathcal{Y}$ interval.

Attack Step I: recover the ideal $\langle g \rangle$. Step I borrows the idea from [CGH17, Section 3.1]. We select a 3-partition $[\ell] = \mathcal{X} \mid \mathcal{Y} \mid \mathcal{Z}$ such that $|\mathcal{Y}| = 2$, $|\mathcal{X}|, |\mathcal{Z}| \geq \log(\rho)$. The r^{th} input bit does not live in $\mathcal{Y}$ and is fixed at 0. We set $y^{(1)} = 00$, $y^{(2)} = 10$, $y^{(3)} = 01$, $y^{(4)} = 11$. Then form a matrix so that computing its determinant reveals an element in $\langle g \rangle$. Note that this step does not concern whether the underlying program is $\Gamma^{(0)}$ or $\Gamma^{(1)}$ since we fix the r^{th} bit of input u as 0.

Algorithm A.3 (Step I)**.** Given the matrices $\{\mathbf{C}_{i,b}\}_{i\in[1,h],b\in\{0,1\}}$, the algorithm proceeds as follows

1. Select a 3-partition $[\ell] = \mathcal{X} \mid \mathcal{Y} \mid \mathcal{Z}$ such that $|\mathcal{Y}| = 2$, $\mathcal{X} = [1, \nu]$, $\mathcal{Z} = [\mu, \ell]$, $|\mathcal{X}|, |\mathcal{Z}| \geq \log(\rho)$. We assume the r^{th} bit lives in $\mathcal{Z}$ and is fixed as 0.
2. Randomly pick ρ prefixes $\left\{x^{(i)} \in \mathcal{X}\right\}_{i\in[\rho]}$, ρ suffixes $\left\{z^{(k)} \in \mathcal{Z}\right\}_{k\in[\rho]}$.
3. Set $y^{(1)} = 00$, $y^{(2)} = 10$, $y^{(3)} = 01$, $y^{(4)} = 11$.
4. For $j \in [4]$: run Alg. A.1 on $\{\mathbf{C}_{i,b}\}_{i\in[1,h],b\in\{0,1\}}$, $\left\{x^{(i)}\right\}_{i\in[\rho]}$, $y^{(j)}$, $\left\{z^{(k)}\right\}_{k\in[\rho]}$, denote the output as $\mathbf{V}^{(j)}$.
5. Compute $\mathbf{W}_N := \begin{pmatrix} \mathbf{V}^{(1)} & \mathbf{V}^{(2)} \\ \mathbf{V}^{(3)} & \mathbf{V}^{(4)} \end{pmatrix}$; $\mathbf{W}_D := \begin{pmatrix} \mathbf{V}^{(1)} & \mathbf{0} \\ \mathbf{0} & \mathbf{V}^{(4)} \end{pmatrix} \in L^{2\rho\times 2\rho}$;
6. Compute $\mathbf{W} := \mathbf{W}_N \cdot \mathbf{W}_D^{-1} \in K^{2\rho\times 2\rho}$.
7. Compute $d = \det(\mathbf{W}) \in K$, output its numerator as $d_N \in L$.

Claim A.4. Let n' be the dimension of the basis of $\mathcal{I}$. Let $\{d_\tau\}_{\tau\in[O(n')]}$ be the outputs by running Alg. A.3 for $O(n')$ times. We claim that each of d_τ is in the ideal $\mathcal{I}$, therefore taking the gcd of $\{d_\tau\}_{\tau\in[O(n')]}$ it is likely to recover a basis of $\mathcal{I}$.

Justification of Claim A.4. We analyze the matrices $\mathbf{W}_N$, $\mathbf{W}_D$

$$\mathbf{W}_N = \begin{pmatrix} \mathbf{X}\mathbf{Y}^{(1)}\mathbf{Z} & \mathbf{X}\mathbf{Y}^{(2)}\mathbf{Z} \\ \mathbf{X}\mathbf{Y}^{(3)}\mathbf{Z} & \mathbf{X}\mathbf{Y}^{(4)}\mathbf{Z} \end{pmatrix} = \begin{pmatrix} \mathbf{X} & \mathbf{0} \\ \mathbf{0} & \mathbf{X} \end{pmatrix} \cdot \begin{pmatrix} \mathbf{Y}^{(1)} & \mathbf{Y}^{(2)} \\ \mathbf{Y}^{(3)} & \mathbf{Y}^{(4)} \end{pmatrix} \cdot \begin{pmatrix} \mathbf{Z} & \mathbf{0} \\ \mathbf{0} & \mathbf{Z} \end{pmatrix} \tag{48}$$

$$\mathbf{W}_D = \begin{pmatrix} \mathbf{X}\mathbf{Y}^{(1)}\mathbf{Z} & \mathbf{0} \\ \mathbf{0} & \mathbf{X}\mathbf{Y}^{(4)}\mathbf{Z} \end{pmatrix} = \begin{pmatrix} \mathbf{X} & \mathbf{0} \\ \mathbf{0} & \mathbf{X} \end{pmatrix} \cdot \begin{pmatrix} \mathbf{Y}^{(1)} & \mathbf{0} \\ \mathbf{0} & \mathbf{Y}^{(4)} \end{pmatrix} \cdot \begin{pmatrix} \mathbf{Z} & \mathbf{0} \\ \mathbf{0} & \mathbf{Z} \end{pmatrix} \tag{49}$$

Following Claim A.2, whp $\begin{pmatrix} \mathbf{X} & \mathbf{0} \\ \mathbf{0} & \mathbf{X} \end{pmatrix}$, $\begin{pmatrix} \mathbf{Z} & \mathbf{0} \\ \mathbf{0} & \mathbf{Z} \end{pmatrix}$ are non-singular over L, therefore $\mathbf{W}_D$ is non-singular. Hence we can continue with $\mathbf{W} := \mathbf{W}_N \cdot \mathbf{W}_D^{-1}$.

$$\begin{aligned} \mathbf{W} &= \begin{pmatrix} \mathbf{X} & \mathbf{0} \\ \mathbf{0} & \mathbf{X} \end{pmatrix} \cdot \begin{pmatrix} \mathbf{Y}^{(1)} & \mathbf{Y}^{(2)} \\ \mathbf{Y}^{(3)} & \mathbf{Y}^{(4)} \end{pmatrix} \cdot \begin{pmatrix} \mathbf{Z} & \mathbf{0} \\ \mathbf{0} & \mathbf{Z} \end{pmatrix} \cdot \left(\begin{pmatrix} \mathbf{X} & \mathbf{0} \\ \mathbf{0} & \mathbf{X} \end{pmatrix} \cdot \begin{pmatrix} \mathbf{Y}^{(1)} & \mathbf{0} \\ \mathbf{0} & \mathbf{Y}^{(4)} \end{pmatrix} \cdot \begin{pmatrix} \mathbf{Z} & \mathbf{0} \\ \mathbf{0} & \mathbf{Z} \end{pmatrix} \right)^{-1} \\ &= \begin{pmatrix} \mathbf{X} & \mathbf{0} \\ \mathbf{0} & \mathbf{X} \end{pmatrix} \cdot \begin{pmatrix} \mathbf{Y}^{(1)} & \mathbf{Y}^{(2)} \\ \mathbf{Y}^{(3)} & \mathbf{Y}^{(4)} \end{pmatrix} \cdot \begin{pmatrix} \mathbf{Y}^{(1)} & \mathbf{0} \\ \mathbf{0} & \mathbf{Y}^{(4)} \end{pmatrix}^{-1} \cdot \begin{pmatrix} \mathbf{X} & \mathbf{0} \\ \mathbf{0} & \mathbf{X} \end{pmatrix}^{-1} \end{aligned} \tag{50}$$

Therefore

$$\det(\mathbf{W}) = \det\left(\begin{pmatrix} \mathbf{Y}^{(1)} & \mathbf{Y}^{(2)} \\ \mathbf{Y}^{(3)} & \mathbf{Y}^{(4)} \end{pmatrix} \cdot \begin{pmatrix} \mathbf{Y}^{(1)} & \mathbf{0} \\ \mathbf{0} & \mathbf{Y}^{(4)} \end{pmatrix}^{-1} \right) \tag{51}$$

So far we have been analyzing the properties of matrices over L and reach the conclusion that to compute $\det(\mathbf{W})$, it is enough to understand the $\mathbf{Y}$ components. Now we start analyzing the properties of $\left\{\mathbf{Y}^{(j)}\right\}_{j\in[4]}$ mod $\mathcal{I}$:

$$\mathbf{Y}^{(j)} = \mathbf{C}_{y_2^{(j)}} \otimes \mathbf{I}^{t\times t} \otimes \mathbf{C}_{y_1^{(j)}} = \hat{\mathbf{S}}_{y_2^{(j)}} \otimes \mathbf{I}^{t\times t} \otimes \hat{\mathbf{S}}_{y_1^{(j)}} \pmod{\mathcal{I}} \tag{52}$$

Here $\hat{\mathbf{S}}_{y_2^{(j)}}$ and $\hat{\mathbf{S}}_{y_1^{(j)}}$ are products of the $\hat{\mathbf{S}}$ terms defined in Eqn (20), which means the resulting $\hat{\mathbf{S}}_{y_2^{(j)}}$ and $\hat{\mathbf{S}}_{y_1^{(j)}}$ contain two blocks on the diagonal, one for the "functional branch" and the other for the "dummy branch".

If we analyze the property of $\mathbf{Y}^{(j)}$ in Eqn. (52) faithfully, it will constitute a significant blowup in size of the equations. Instead we will ignore the "dummy branch" and the $\mathbf{I}^{t\times t}$ term for now, and focus on the term

$$\tilde{\mathbf{Y}}^{(j)} := \left(\alpha_{y_2^{(j)}} \cdot \mathbf{K}_{\ell+\nu}^{-1} \cdot \begin{pmatrix} \mathbf{R}_{y_2^{(j)}} & \\ & \mathbf{M}_{y_2^{(j)}} \end{pmatrix} \cdot \mathbf{K}_{\ell+\mu-1}\right) \otimes \left(\alpha_{y_1^{(j)}} \cdot \mathbf{K}_{\nu}^{-1} \cdot \begin{pmatrix} \mathbf{R}_{y_1^{(j)}} & \\ & \mathbf{M}_{y_1^{(j)}} \end{pmatrix} \cdot \mathbf{K}_{\mu-1}\right) \tag{53}$$

The readers can verify that $\tilde{\mathbf{Y}}^{(j)}$ can be set as a diagonal block of $\mathbf{Y}^{(j)}$ after a few elementary linear operations, i.e.

$$\mathbf{Y}^{(j)} = \mathbf{U}_L \cdot \begin{pmatrix} \tilde{\mathbf{Y}}^{(j)} & \\ & *^{(j)} \end{pmatrix} \cdot \mathbf{U}_R \pmod{\mathcal{I}}. \tag{54}$$

where $\mathbf{U}_L, \mathbf{U}_R$ are elementary matrices that do not depend on j.

We claim that the analyses of $\tilde{\mathbf{Y}}^{(j)}$ below suffice to justify Claim A.4. Now we dig into $\tilde{\mathbf{Y}}^{(j)}$:

$$\tilde{\mathbf{Y}}^{(j)} = \alpha_{y_2^{(j)}} \cdot \alpha_{y_1^{(j)}} \cdot \underbrace{\left(\mathbf{K}_{\ell+\nu}^{-1} \otimes \mathbf{K}_{\nu}^{-1}\right)}_{:=\tilde{\mathbf{K}}_{\nu}^{-1}} \cdot \underbrace{\left(\begin{pmatrix} \mathbf{R}_{y_2^{(j)}} & \\ & \mathbf{I}^{w\times w} \end{pmatrix} \otimes \begin{pmatrix} \mathbf{R}_{y_1^{(j)}} & \\ & \mathbf{I}^{w\times w} \end{pmatrix}\right)}_{:=\begin{pmatrix} \tilde{\mathbf{R}}_{y^{(j)}} & \\ & \mathbf{I}^{w\times w} \end{pmatrix} \in \begin{pmatrix} R^{(t^2-w)\times(t^2-w)} & \\ & \mathbf{I}^{w\times w} \end{pmatrix}} \cdot \underbrace{\left(\mathbf{K}_{\ell+\mu-1} \otimes \mathbf{K}_{\mu-1}\right)}_{:=\tilde{\mathbf{K}}_{\mu-1}} \tag{55}$$

Putting together all the 4 pieces $\left\{\tilde{\mathbf{Y}}^{(j)}\right\}_{j\in[4]}$ gives

$$\begin{aligned}
\tilde{\mathbf{Y}} &:= \begin{pmatrix} \tilde{\mathbf{Y}}^{(1)} & \tilde{\mathbf{Y}}^{(2)} \\ \tilde{\mathbf{Y}}^{(3)} & \tilde{\mathbf{Y}}^{(4)} \end{pmatrix} \\
&= \begin{pmatrix} \alpha_{y_2^{(1)}} \cdot \alpha_{y_1^{(1)}} \cdot \tilde{\mathbf{K}}_{\nu}^{-1} \cdot \begin{pmatrix} \tilde{\mathbf{R}}_{y^{(1)}} & \\ & \mathbf{I}^{w\times w} \end{pmatrix} \cdot \tilde{\mathbf{K}}_{\mu-1} & \alpha_{y_2^{(2)}} \cdot \alpha_{y_1^{(2)}} \cdot \tilde{\mathbf{K}}_{\nu}^{-1} \cdot \begin{pmatrix} \tilde{\mathbf{R}}_{y^{(2)}} & \\ & \mathbf{I}^{w\times w} \end{pmatrix} \cdot \tilde{\mathbf{K}}_{\mu-1} \\ \alpha_{y_2^{(3)}} \cdot \alpha_{y_1^{(3)}} \cdot \tilde{\mathbf{K}}_{\nu}^{-1} \cdot \begin{pmatrix} \tilde{\mathbf{R}}_{y^{(3)}} & \\ & \mathbf{I}^{w\times w} \end{pmatrix} \cdot \tilde{\mathbf{K}}_{\mu-1} & \alpha_{y_2^{(4)}} \cdot \alpha_{y_1^{(4)}} \cdot \tilde{\mathbf{K}}_{\nu}^{-1} \cdot \begin{pmatrix} \tilde{\mathbf{R}}_{y^{(4)}} & \\ & \mathbf{I}^{w\times w} \end{pmatrix} \cdot \tilde{\mathbf{K}}_{\mu-1} \end{pmatrix} \\
&= \begin{pmatrix} \tilde{\mathbf{K}}_{\nu}^{-1} & \\ & \tilde{\mathbf{K}}_{\nu}^{-1} \end{pmatrix} \cdot \underbrace{\begin{pmatrix} \beta_{\nu+1,0}\beta_{\nu+2,0} \cdot \begin{pmatrix} \tilde{\mathbf{R}}_{y^{(1)}} & \\ & \mathbf{I}^{w\times w} \end{pmatrix} & \beta_{\nu+1,1}\beta_{\nu+2,0} \cdot \begin{pmatrix} \tilde{\mathbf{R}}_{y^{(2)}} & \\ & \mathbf{I}^{w\times w} \end{pmatrix} \\ \beta_{\nu+1,0}\beta_{\nu+2,1} \cdot \begin{pmatrix} \tilde{\mathbf{R}}_{y^{(3)}} & \\ & \mathbf{I}^{w\times w} \end{pmatrix} & \beta_{\nu+1,1}\beta_{\nu+2,1} \cdot \begin{pmatrix} \tilde{\mathbf{R}}_{y^{(4)}} & \\ & \mathbf{I}^{w\times w} \end{pmatrix} \end{pmatrix}}_{:=\mathbf{Q}} \cdot \begin{pmatrix} \tilde{\mathbf{K}}_{\mu-1} & \\ & \tilde{\mathbf{K}}_{\mu-1} \end{pmatrix}
\end{aligned} \tag{56}$$

where the third equality follows Eqn. (19).

Claim A.5. $\det(\mathbf{Q}) = 0 \pmod{\mathcal{I}}$, therefore $\det(\tilde{\mathbf{Y}}) = 0 \pmod{\mathcal{I}}$, therefore $\det(\mathbf{W}) = 0 \pmod{\mathcal{I}}$.

Justification of Claim A.5. Taking the following $2w \times 2w$ blocks out of $\mathbf{Q}$:

$$\mathbf{Q}_\beta := \begin{pmatrix} \beta_{\nu+1,0}\beta_{\nu+2,0}\mathbf{I}^{w\times w} & \beta_{\nu+1,1}\beta_{\nu+2,0}\mathbf{I}^{w\times w} \\ \beta_{\nu+1,0}\beta_{\nu+2,1}\mathbf{I}^{w\times w} & \beta_{\nu+1,1}\beta_{\nu+2,1}\mathbf{I}^{w\times w} \end{pmatrix} \tag{57}$$

Given that $\mathbf{Q}_\beta$ is singular, so $\det(\mathbf{Q}) = 0 \pmod{\mathcal{I}}$. Therefore $\det(\tilde{\mathbf{Y}}) = 0 \pmod{\mathcal{I}}$.

Then $\begin{pmatrix} \mathbf{Y}^{(1)} & \mathbf{Y}^{(2)} \\ \mathbf{Y}^{(3)} & \mathbf{Y}^{(4)} \end{pmatrix}$ is singular mod $\mathcal{I}$ since $\tilde{\mathbf{Y}}$ can be set to its diagonal block after elementary linear operations, following Eqn. (54).

Finally we observe that $\begin{pmatrix} \mathbf{Y}^{(1)} & \mathbf{0} \\ \mathbf{0} & \mathbf{Y}^{(4)} \end{pmatrix}$ is non-singular both mod $\mathcal{I}$ and in L, while $\det \begin{pmatrix} \mathbf{Y}^{(1)} & \mathbf{Y}^{(2)} \\ \mathbf{Y}^{(3)} & \mathbf{Y}^{(4)} \end{pmatrix}$ is singular mod $\mathcal{I}$ but non-singular in L. Therefore $\det(\mathbf{W})$ is an non-zero element in $\mathcal{I}$. This concludes the verification of Claim A.5.

Now Claim A.4 follows by repeating Alg. A.3 several times with different partition boundaries, then we can take the GCD of the resulting $\det(\mathbf{W})$ values and whp get a basis for $\mathcal{I}$.

Note that if we follow strictly from Alg. A.3, what we get in the reality is $\mathcal{I}^d$ for some polynomial d (which means the value of d can be guessed in polynomial time). This can be fixed by 2 methods (1) Computing $\det(\tilde{\mathbf{W}})$ instead of $\det(\mathbf{W})$, where $\tilde{\mathbf{W}}$ is a $(2\rho - d + 1)$-by-$(2\rho - d + 1)$ block of $\mathbf{W}$. Our experiment indicates that $\det(\tilde{\mathbf{W}})$ lives exactly in $\mathcal{I}$. (2) When $\mathcal{I}$ is a prime ideal, it is easy to find $\mathcal{I}$ from $\mathcal{I}^d$: The norm of $\mathcal{I}^d$ is $\mathrm{norm}(\mathcal{I})^{d'}$, and $p = \mathrm{norm}(\mathcal{I})$ is a prime integer, and we can find p from p^d. The Kummer-Dedekind theorem let us compute all the ideals of norm p in K, and one of these ideals is $\mathcal{I}$.

Attack Step II: distinguishing $\Gamma^{(0)}$ and $\Gamma^{(1)}$. Step II still follows the idea from [CGH17, Section 3.2], but this time we use a feature from the characteristic polynomial of some matrix to distinguish $\Gamma^{(0)}$ and $\Gamma^{(1)}$ directly, instead of getting the bundling scalars from the roots of the characteristic polynomial and use them in another painful step.

Algorithm A.6 (Step II)**.** Given the obfuscated code $\{\mathbf{C}_{i,b}\}_{i\in[1,h],b\in\{0,1\}}$ of one of $\Gamma^{(0)}$, $\Gamma^{(1)}$, a basis of $\mathcal{I}$ obtained in Step I, the algorithm proceeds as follows

1. Select a 3-partition $[\ell] = \mathcal{X} \mid \mathcal{Y} \mid \mathcal{Z}$ such that $\mathcal{Y}$ contains only the r^{th} input bit (therefore $\mathcal{X}$ and $\mathcal{Z}$ are determined).
2. Randomly pick ρ prefixes $\left\{x^{(i)} \in \mathcal{X}\right\}_{i\in[\rho]}$, ρ suffixes $\left\{z^{(k)} \in \mathcal{Z}\right\}_{k\in[\rho]}$.
3. Set $y^{(5)} = 0$, $y^{(6)} = 1$.
4. For $j \in \{5, 6\}$: run Alg. A.1 on $\{\mathbf{C}_{i,b}\}_{i\in[1,h],b\in\{0,1\}}$, ρ, $\left\{x^{(i)}\right\}_{i\in[\rho]}$, $y^{(j)}$, $\left\{z^{(k)}\right\}_{k\in[\rho]}$, denote the output as $\mathbf{V}^{(j)}$.
5. Compute the characteristic polynomial of $\mathbf{V}^{(6)} \cdot \left(\mathbf{V}^{(5)}\right)^{-1} \pmod{\mathcal{I}}$, factorize the polynomial. If there exists a factor of multiplicity $\geq 2w^2 \cdot t$, then it is $\Gamma^{(0)}$; otherwise it is $\Gamma^{(1)}$.

Analysis. Let $\chi(\mathbf{M})$ denote the characteristic polynomial of a square matrix $\mathbf{M}$. We have

$$\begin{aligned}
\chi\left(\mathbf{V}^{(6)}\cdot\left(\mathbf{V}^{(5)}\right)^{-1}\right) &= \chi\left(\mathbf{X}\cdot\mathbf{Y}^{(6)}\cdot\mathbf{Z}\cdot\left(\mathbf{X}\cdot\mathbf{Y}^{(5)}\cdot\mathbf{Z}\right)^{-1}\right)\\
&= \chi\left(\mathbf{X}\cdot\mathbf{Y}^{(6)}\cdot\left(\mathbf{Y}^{(5)}\right)^{-1}\cdot\mathbf{X}^{-1}\right)\\
&= \chi\left(\mathbf{Y}^{(6)}\cdot\left(\mathbf{Y}^{(5)}\right)^{-1}\right)\\
&= \chi\left(\left(\mathbf{C}_{y_2^{(6)}}\otimes\mathbf{I}^{t\times t}\otimes\mathbf{C}_{y_1^{(6)}}\right)\cdot\left(\mathbf{C}_{y_2^{(5)}}\otimes\mathbf{I}^{t\times t}\otimes\mathbf{C}_{y_1^{(5)}}\right)^{-1}\right)\\
&= \chi\left(\left(\hat{\mathbf{S}}_{y_2^{(6)}}\otimes\mathbf{I}^{t\times t}\otimes\hat{\mathbf{S}}_{y_1^{(6)}}\right)\cdot\left(\hat{\mathbf{S}}_{y_2^{(5)}}\otimes\mathbf{I}^{t\times t}\otimes\hat{\mathbf{S}}_{y_1^{(5)}}\right)^{-1}\right) \pmod{\mathcal{I}}\\
&= \chi\left(\left(\hat{\mathbf{S}}_{y_2^{(6)}}\cdot\left(\hat{\mathbf{S}}_{y_2^{(5)}}\right)^{-1}\right)\otimes\mathbf{I}^{t\times t}\otimes\left(\hat{\mathbf{S}}_{y_1^{(6)}}\cdot\left(\hat{\mathbf{S}}_{y_1^{(5)}}\right)^{-1}\right)\right)
\end{aligned} \tag{58}$$

where the 1^{st} and the 4^{th} equalities follow Eqn. (45).

For square matrices $\mathbf{A}$, $\mathbf{B}$, the set of eigenvalues of $\mathbf{A}\otimes\mathbf{B}$ is the set of products (with multiplicities) of eigenvalues of $\mathbf{A}$ with those of $\mathbf{B}$. We first analyze the eigenvalues of $\hat{\mathbf{S}}_{y_1^{(6)}}\cdot\left(\hat{\mathbf{S}}_{y_1^{(5)}}\right)^{-1}$:

$$\begin{aligned}
&\chi\left(\hat{\mathbf{S}}_{y_1^{(6)}}\cdot\left(\hat{\mathbf{S}}_{y_1^{(5)}}\right)^{-1}\right) = \chi\left(\hat{\mathbf{S}}_{r,1}\cdot\left(\hat{\mathbf{S}}_{r,0}\right)^{-1}\right)\\
&= \chi\left(\mathrm{diag}\left(\alpha_{r,1}/\alpha_{r,0}\cdot\mathbf{K}_{r-1}^{-1}\begin{pmatrix}\mathbf{M}_{r,1}\cdot\mathbf{M}_{r,0}^{-1} & \\ & \mathbf{R}_{r,1}\cdot\mathbf{R}_{r,0}^{-1}\end{pmatrix}\mathbf{K}_{r-1}\ ,\ \alpha'_{r,1}/\alpha'_{r,0}\cdot\mathbf{K'}_{r-1}^{-1}\begin{pmatrix}\mathbf{I} & \\ & \mathbf{R}'_{r,1}\cdot\mathbf{R'}_{r,0}^{-1}\end{pmatrix}\mathbf{K'}_{r-1}\right)\right)\\
&= \chi\left(\alpha_{r,1}/\alpha_{r,0}\cdot\begin{pmatrix}\mathbf{M}_{r,1}\cdot\mathbf{M}_{r,0}^{-1} & \\ & \mathbf{R}_{r,1}\cdot\mathbf{R}_{r,0}^{-1}\end{pmatrix}\right)\cdot\chi\left(\alpha'_{r,1}/\alpha'_{r,0}\cdot\begin{pmatrix}\mathbf{I} & \\ & \mathbf{R}'_{r,1}\cdot\mathbf{R'}_{r,0}^{-1}\end{pmatrix}\right)
\end{aligned} \tag{59}$$

Now we observe that $\alpha'_{r,1}/\alpha'_{r,0}$ is always an eigenvalue of $\hat{\mathbf{S}}_{y_1^{(6)}}\cdot\left(\hat{\mathbf{S}}_{y_1^{(5)}}\right)^{-1}$ with multiplicity w. The multiplicity of $\alpha_{r,1}/\alpha_{r,0}$ depends on the underlying program Γ.

- If the underlying program is $\Gamma^{(0)}$, then $\mathbf{M}_{r,1}\cdot\mathbf{M}_{r,0}^{-1}=\mathbf{I}^{w\times w}$, then $\alpha_{r,1}/\alpha_{r,0}$ is an eigenvalue of $\hat{\mathbf{S}}_{y_1^{(6)}}\cdot\left(\hat{\mathbf{S}}_{y_1^{(5)}}\right)^{-1}$ with multiplicity w.
- If the underlying program is $\Gamma^{(1)}$, then $\mathbf{M}_{r,1}\cdot\mathbf{M}_{r,0}^{-1}=\mathbf{N}\neq\mathbf{I}$, then $\alpha_{r,1}/\alpha_{r,0}$ is an eigenvalue of $\hat{\mathbf{S}}_{y_1^{(6)}}\cdot\left(\hat{\mathbf{S}}_{y_1^{(5)}}\right)^{-1}$ with multiplicity $< w$.

The similar argument can be made on $\hat{\mathbf{S}}_{y_2^{(6)}}\cdot\left(\hat{\mathbf{S}}_{y_2^{(5)}}\right)^{-1}$.

When we put everything together, in the calculation of final threshold multiplicity, one has to take into account the correlations among the scalars. Following Eqn. (19), $\beta_{r,b}=\alpha_{r,b}\alpha_{\ell+r,b}=\alpha'_{r,b}\alpha'_{\ell+r,b}$, $b\in\{0,1\}$. So if the underlying program is $\Gamma^{(0)}$, then $\beta_{r,1}/\beta_{r,0}$ is an eigenvalue of $\mathbf{V}^{(6)}\cdot\left(\mathbf{V}^{(5)}\right)^{-1}$ with multiplicity $2w^2\cdot t$. If the underlying program is $\Gamma^{(1)}$, then there is no eigenvalue of $\mathbf{V}^{(6)}\cdot\left(\mathbf{V}^{(5)}\right)^{-1}$ with multiplicity $\geq 2w^2\cdot t$.

Extension to c-input-repeating BPs. For single input oblivious branching programs with c-input-repetitions (i.e. $h=c\cdot\ell$), the attack still applies when $c>2$. However, ρ (the dimension of a matrix in the subroutine) grows exponentially with c when converting the evaluation formula into 3-partition by recursively applying Formula (34). So the running time of the attack is $\mathrm{poly}(t)^c=\mathrm{poly}(\lambda,\ell)^c$ where λ is the security parameter, ℓ is the input length.