Matrix PRFs: Constructions, Attacks, and Applications to Obfuscation

Yilei Chen
Visa Research*

Minki Hhan
SNU†

Vinod Vaikuntanathan
MIT CSAIL‡

Hoeteck Wee
CNRS, ENS, PSL§

September 23, 2019

Abstract

We initiate a systematic study of pseudorandom functions (PRFs) that are computable by simple matrix branching programs; we refer to these objects as "matrix PRFs". Matrix PRFs are attractive due to their simplicity, strong connections to complexity theory and group theory, and recent applications in program obfuscation.

Our main results are:

- We present constructions of matrix PRFs based on the conjectured hardness of computational problems pertaining to matrix products.
- We show that any matrix PRF that is computable by a read-c, width w branching program can be broken in time poly(w^c); this means that any matrix PRF based on constant-width matrices must read each input bit $\omega(\log(\lambda))$ times. Along the way, we simplify the "tensor switching lemmas" introduced in previous IO attacks.
- We show that a subclass of the candidate local-PRG proposed by Barak et al. [Eurocrypt 2018] can be broken using simple matrix algebra.
- We show that augmenting the CVW18 IO candidate with a matrix PRF provably immunizes the candidate against all known algebraic and statistical zeroizing attacks, as captured by a new and simple adversarial model.

*E-mail: chenyilei.ra@gmail.com. Research conducted while the author was at Boston University supported by the NSF MACS project and NSF grant CNS-1422965.

†E-mail: hhan_@snu.ac.kr. Research supported by Institute for Information & communication Technology Promotion (IITP) grant funded by the Korea government (MSIT) (No. 2016-6-00598, The mathematical structure of functional encryption and its analysis), and the ARO and DARPA under Contract No. W911NF-15-C-0227.

‡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).

Contents

1 Introduction

Pseudorandom functions (PRFs), defined by Goldreich, Goldwasser, and Micali [29], are keyed functions that are indistinguishable from truly random functions given black-box access. In this work we focus on pseudorandom functions that can be represented by simple matrix branching programs; we refer to these objects as "matrix PRFs". In the simplest setting, a matrix PRF takes a key specified by ℓ pairs of $w \times w$ matrices $\{\mathbf{M}_{i,b}\}_{i\in[\ell],b\in\{0,1\}}$ where

$$\mathsf{PRF}(\{\mathbf{M}_{i,b}\}_{i\in[\ell],b\in\{0,1\}}, \mathbf{x} \in \{0,1\}^{\ell}) := \prod_{i=1}^{\ell} \mathbf{M}_{i,x_i}$$

Matrix PRFs are attractive due to their simplicity, strong connections to complexity theory and group theory [12, 44, 1], and recent applications in program obfuscation [27, 11].

Existing constructions. First, we note that the Naor-Reingold PRF [37] (extended to matrices in [34]) and the Banerjee-Peikert-Rosen PRF [7] may be viewed as matrix PRFs with post-processing, corresponding to group exponentiation and entry-wise rounding respectively. However, the applications we have in mind do not allow such post-processing. Instead, we turn to a more general definition of read-c matrix PRFs, where the key is specified by $h := c \cdot \ell$ pairs of $w \times w$ matrices $\{\mathbf{M}_{i,b}\}_{i\in[h],b\in\{0,1\}}$ where

$$\mathsf{PRF}(\{\mathbf{M}_{i,b}\}_{i\in[h],b\in\{0,1\}}, \mathbf{x}) := \mathbf{u}_L \cdot \prod_{i=1}^{h} \mathbf{M}_{i,x_{i \bmod \ell}} \cdot \mathbf{u}_R \tag{1}$$

Here, $\mathbf{u}_L, \mathbf{u}_R$ correspond to fixed vectors independent of the key. This corresponds exactly to PRFs computable by read-c matrix branching programs. By applying Barrington's theorem on the existing PRFs in NC^1, such as the two PRFs we just mentioned [37, 7], we obtain read-poly(ℓ) matrix PRFs based on standard assumptions like DDH and LWE.

This work. In this work, we initiate a systematic study of matrix PRFs.

- From the constructive perspective, we investigate whether there are "simpler" constructions of matrix PRF, or hardness assumptions over matrix products that can be used to build matrix PRFs. Here "simpler" means the matrices $\mathbf{M}_{i,b}$'s are drawn from some "natural" distribution, for instance, independently at random from the same distribution. Note that the constructions obtained by apply Barrington's theorem [10] on PRFs in NC^1 yield highly correlated and structured distributions.

- From the attacker's perspective, the use of matrices opens the gate for simple linear algebraic attacks in breaking the hardness assumptions. We would like to understand what are the characteristics that a matrix PRF could or could not have, by trying different linear algebraic attacks. These characteristics include the distribution of the underlying matrices, as well as the complexity of the underlying branching program.

- Finally, we revisit the application of matrix PRFs to program obfuscation as a mechanism for immunizing against known attacks.

1.1 Our Contributions

Our contributions may be broadly classified into three categories, corresponding to the three lines of questions mentioned above.

1.1.1 Attacks

We show that any matrix PRF that is computable by a read-c, width-w branching program can be broken in time poly(w^c); this means that any matrix PRF based on constant-width matrices must read each input bit $\omega(\log(\lambda))$ times. Our attack and the analysis are inspired by previous zeroizing attacks on obfuscation [23, 6, 18]; we also provide some simplification along the way. We note that the case of $c = 1$ appears to be folklore.

The attack. The attack is remarkably simple: given oracle access to a function $F : \{0,1\}^\ell \to R$,

1. pick any $L := w^{2c}$ distinct strings $x_1, \ldots, x_L \in \{0,1\}^{\ell/2}$;
2. compute $\mathbf{V} \in R^{L \times L}$ whose (i,j)'th entry is $F(x_i \| x_j)$;
3. output $\mathsf{rank}(\mathbf{V})$

If F is a truly random function, then $\mathbf{V}$ has full rank w.h.p. On the other hand, if F is computable by a read-c, width w branching program, then we show that $F(x_i \| x_j)$ can be written in the form $\langle \mathbf{u}_i, \mathbf{v}_j \rangle$ for some fixed $\mathbf{u}_1, \ldots, \mathbf{u}_L, \mathbf{v}_1, \ldots, \mathbf{v}_L \in R^{w^{2c-1}}$. This means that we can write

$$\mathbf{V} = \underbrace{\begin{pmatrix} \leftarrow \mathbf{u}_1 \rightarrow \\ \vdots \\ \leftarrow \mathbf{u}_L \rightarrow \end{pmatrix}}_{L \times w^{2c-1}} \underbrace{\begin{pmatrix} \uparrow & & \uparrow \\ \mathbf{v}_1 & \cdots & \mathbf{v}_L \\ \downarrow & & \downarrow \end{pmatrix}}_{w^{2c-1} \times L}$$

which implies $\mathsf{rank}(\mathbf{V}) \leq w^{2c-1}$.

Next, we sketch how we can decompose $F(x_i \| x_j)$ into $\langle \mathbf{u}_i, \mathbf{v}_j \rangle$. This was already shown in [23, Section 4.2], but we believe our analysis is simpler and more intuitive. Consider a read-thrice branching program of width w where

$$\mathbf{M}_{x\|y} = \mathbf{u}_L \mathbf{M}_x^1 \mathbf{N}_y^1 \mathbf{M}_x^2 \mathbf{N}_y^2 \mathbf{M}_x^3 \mathbf{N}_y^3 \mathbf{u}_R$$

Suppose we can rewrite $\mathbf{M}_{x\|y}$ as

$$\begin{aligned} & \hat{\mathbf{u}}_L \cdot (\mathbf{M}_x^1 \mathbf{N}_y^1) \otimes (\mathbf{M}_x^2 \mathbf{N}_y^2) \otimes (\mathbf{M}_x^3 \mathbf{N}_y^3)) \cdot \hat{\mathbf{u}}_R \\ = \; & \underbrace{\hat{\mathbf{u}}_L \cdot (\mathbf{M}_x^1 \otimes \mathbf{M}_x^2 \otimes \mathbf{M}_x^3)}_{1 \times w^3} \cdot \underbrace{(\mathbf{N}_y^1 \otimes \mathbf{N}_y^2 \otimes \mathbf{N}_y^3) \cdot \hat{\mathbf{u}}_R}_{w^3 \times 1} \end{aligned}$$

for some suitable choices of $\hat{\mathbf{u}}_L, \hat{\mathbf{u}}_R$. Unfortunately, such a statement appears to be false. Nonetheless, we are able to prove a similar decomposition where we replace $\hat{\mathbf{u}}_L \cdot (\mathbf{M}_x^1 \otimes \mathbf{M}_x^2 \otimes \mathbf{M}_x^3)$ on the left with

$$\underbrace{\mathsf{flat}\big(\overbrace{\mathbf{u}_L \mathbf{M}_x^1 \otimes \mathbf{M}_x^2 \otimes \mathbf{M}_x^3}^{w^2 \times w^3}\big)}_{1 \times w^5}$$

where flat "flattens" a $n \times m$ matrix into a $1 \times nm$ row vector by concatenating the rows of the input matrix.

1.1.2 Constructions

We show how to build matrix PRFs starting from elementary assumptions over matrix products. Concretely, one of the assumptions is

$$\left(\{\mathbf{A}_{i,b}\}_{i\in[k],b\in\{0,1\}}, \prod_{i=1}^{k}(\mathbf{A}_{i,0}\mathbf{B}), \prod_{i=1}^{k}(\mathbf{A}_{i,1}\mathbf{B})\right) \approx_c \left(\{\mathbf{A}_{i,b}\}_{i\in[k],b\in\{0,1\}}, \mathbf{B}_0, \mathbf{B}_1\right) \tag{2}$$

where the matrices $\mathbf{A}_{i,b}$, $\mathbf{B}$, $\mathbf{B}_0$ and $\mathbf{B}_1$ are sampled uniformly random over some non-abelian simple groups. From there a matrix PRF can be obtained via the Naor-Reingold paradigm [37]. We also provide another assumption over matrix products and a corresponding synthesizer-style matrix PRF.

Both of the matrix PRFs mentioned above are based on elementary assumptions over matrix products and are polynomial-time computable, but the evaluation results are products of super-polynomially many matrices when taking inputs of polynomially many bits. In other words, they cannot be written as polynomially long matrix branching programs. To this end, let us mention that we also conjecture the simple subset product-based construction á la Eqn. (1), even without the bookends, is a PRF when the matrices are sampled from a non-commutative simple group $\mathbb{G}$, and each input is repeated for at least λ times (i.e., $c \in O(\lambda)$). But we do not know how to base its pseudorandomness from an elementary assumption.

1.1.3 Applications to IO

We show that augmenting the CVW18 GGH15-based IO candidate with a matrix PRF provably immunizes the candidate against known algebraic and statistical zeroizing attacks, as captured by a new and simple adversarial model.

Our IO candidate. Our IO candidate on a branching program for a function $f : \{0,1\}^\ell \to \{0,1\}$ samples random Gaussian matrices $\{\mathbf{S}_{i,b}\}_{i\in[h],b\in\{0,1\}}$, a random vector $\mathbf{a}_h$ over $\mathbb{Z}_q$ and a random matrix PRF $\mathsf{PRF}_{\mathbf{M}} : \{0,1\}^\ell \to [0, 2^\tau]$ where $2^\tau \ll q$, and outputs

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

The construction basically follows that in [18], with the matrix PRF embedded along the diagonal. By padding the programs, we may assume that the input program and the matrix PRF share the same input-to-index function $\varpi : \{0,1\}^h \to \{0,1\}^\ell$. Then, we have

$$\mathbf{A}_J\mathbf{D}_{\varpi(\mathbf{x})} \bmod q \approx \begin{cases} 0 \cdot \mathbf{S}_{\varpi(\mathbf{x})}\mathbf{a}_h + \mathsf{PRF}_{\mathbf{M}}(\mathbf{x}) & \text{if } f(\mathbf{x}) = 1 \\ (\neq 0) \cdot \mathbf{S}_{\varpi(\mathbf{x})}\mathbf{a}_h + \mathsf{PRF}_{\mathbf{M}}(\mathbf{x}) & \text{if } f(\mathbf{x}) = 0 \end{cases}$$

where $\approx$ captures an error term which is much smaller than 2^τ. Functionality is straight-forward: output 1 if $\|\mathbf{A}_J\mathbf{D}_{\varpi(\mathbf{x})}\| < 2^\tau$ and 0 otherwise.

Our attack model. We introduce the *input-consistent evaluation model* on GGH15-based IO candidates, where the adversary gets oracle access to

$$O_r(\mathbf{x}) := \mathbf{A}_J \mathbf{D}_{\varpi(\mathbf{x})} \bmod q$$

instead of $\mathbf{A}_J, \{\mathbf{D}_{i,b}\}_{i\in[h],b\in\{0,1\}}$. Basically, all known attacks on GGH15-based IO candidates (including the rank attack and statistical zeroizing attacks [18, 19] can be implemented in this model. In fact, many of these attacks only make use of the low-norm quantities $\{O_r(\mathbf{x}) : f(\mathbf{x}) = 1\}$, which are also referred to as encodings of zeros, and hence the name zeroizing attacks.

Note that our model allow the adversary to perform arbitrary polynomial-time computation on the output of $O_r(\cdot)$, whereas the "weak multi-linear map model" in [11] only allows for algebraic computation of these quantities. The latter does not capture computing the norm of these quantities, as was done in the recent statistical zeroizing attacks [19]. In fact, we even allow the adversary access to $\{\mathbf{A}_J \mathbf{D}_{\varpi(\mathbf{x})} \bmod q : f(\mathbf{x}) = 0\}$, quantities which none of the existing attack takes advantage of except the some attacks [21, 18] for a simple GGH15 obfuscation [31]. In fact, the class of adversaries that only does such evaluations appears to capture all known attacks for GGH15-based obfuscation.

We clarify that our attack model does not capture so-called mixed-input attacks, where the adversary computes $\mathbf{A}_J \mathbf{D}_{\mathbf{x}'} \bmod q$ for some $\mathbf{x}' \notin \varpi(\{0,1\}^\ell)$. As in prior works, we make sure that such quantities do not have small norm, but pre-processing the branching program to reject all $\mathbf{x}' \notin \varpi(\{0,1\}^\ell)$ (see 6.1.2 for details).

Analysis. We show that for our IO candidate, we can simulate oracle access to $O_r(\cdot)$ given oracle access to $f(\cdot)$ under the LWE assumption (which in particular implies the existence of matrix PRFs). This basically says that our IO candidate achieves "virtual black-box security" in the input-consistent evaluation model.

The proof strategy is quite simple: we hide the lower bits by using the embedded matrix PRFs, and hide the higher bits using lattice-based PRFs [7, 14]. In more detail, observe that the lower τ bits of of $O_r(\cdot)$ are pseudorandom, thanks to pseudorandomness of $\mathsf{PRF}_{\mathbf{M}}(\cdot)$. We can then simulate the higher $\log q - \tau$ bits exactly as in [18]:

- if $f(\mathbf{x}) = 1$, then these bits are just 0.
- if $f(\mathbf{x}) = 0$, then we can just rely on the pseudorandomness of existing LWE-based PRFs [14, 7], which tells us that the higher $\log q - \tau$ bits of $\mathbf{S}_{\varpi(\mathbf{x})}\mathbf{a}_h$ are pseudorandom.

Note that the idea of embedding a matrix PRF into an IO candidate already appeared in [27, Section 1.3]; however, the use of matrix PRF for "noise flooding" the encodings of zeros and the lower-order bits as in our analysis –while perfectly natural in hindsight– appears to be novel to this work. In prior works [27, 11], the matrix PRF is merely used to rule out non-trivial algebraic relations amongst the encodings of zeros, namely that there is no low-degree polynomial that vanishes over a large number of pseudorandom values.

1.2 Discussion

Implications for IO. Our results demonstrate new connections between matrix PRFs and IO in this work and shed new insights into existing IO constructions and candidates:

- Many candidates for IO follow the template laid out in [26]: start out with a branching program $\{\mathbf{M}_{i,b}\}_{i\in[h],b\in\{0,1\}}$, perform some pre-processing, and encode the latter using graded encodings. To achieve security in the generic group model [9] or to defeat against the rank attack [18], the pre-processing would add significant redundancy or blow up the length of the underlying branching program. In particular, even if we start out with a read-once branching program as considered in [31], the program we encode would be a read-ℓ (e.g. for so-called dual-input branching programs) or read-λ branching program. But, why read-ℓ or read-λ? Our results –both translating existing IO attacks to attacks on matrix PRFs, and showing how to embed a matrix PRF to achieve resilience against existing attacks– suggest that the blow-up is closely related to the complexity of computing matrix PRFs.

- A recent series of works demonstrated a close connection between building functional encryption (and thus IO) to that of low-degree pseudorandom generators (PRG) over the *integers* [35, 5, 2], where the role of the PRGs is to flood any leakage from the error term during FHE decryption [30]. Here, we show to exploit matrix PRFs –again over the *integers*– to flood any leakage from the error term in the GGH15 encodings (but unlike the setting of PRGs, we do not require the output of the PRFs to have polynomially bounded domain). Both these lines of works point to understanding pseudorandomness over the integers as a crucial step towards building IO.

- Our results suggest new avenues for attacks using input-inconsistent evaluations, namely to carefully exploit the quantities $\{\mathbf{A}_J\mathbf{D}_{\mathbf{x}'} \bmod q : \mathbf{x}' \notin \varpi(\{0,1\}^\ell)\}$ instead of the input-consistent evaluations.

We note that our attacks also play a useful pedagogical role: explaining the core idea of existing zeroizing attacks on IO in the much simpler context of breaking pseudorandomness of matrix PRFs.

Additional related works. Let us remark that recently Boneh et al. [13] also look for (weak) PRFs with simple structures, albeit with a different flavor of simplicity. Their candidates in fact use the change of modulus, which is what we are trying to avoid.

2 Preliminaries

Notations and terminology. Let $\mathbb{R}, \mathbb{Z}, \mathbb{N}$ be the set of real numbers, integers and positive integers. Denote $\mathbb{Z}/(q\mathbb{Z})$ by $\mathbb{Z}_q$. For $n \in \mathbb{N}$, let $[n] := \{1, ..., n\}$. 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 $\mathbf{a}_i$.

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

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{1,1}\mathbf{B}, & \dots, & a_{1,m}\mathbf{B} \\ \dots, & \dots, & \dots \\ a_{\ell,1}\mathbf{B}, & \dots, & a_{\ell,m}\mathbf{B} \end{bmatrix} \in \mathbb{R}^{\ell n\times mp}. \tag{3}$$

For matrices $\mathbf{A} \in \mathbb{R}^{\ell \times m}$, $\mathbf{B} \in \mathbb{R}^{n \times p}$, $\mathbf{C} \in \mathbb{R}^{m \times u}$, $\mathbf{D} \in \mathbb{R}^{p \times v}$,

$$(\mathbf{AC}) \otimes (\mathbf{BD}) = (\mathbf{A} \otimes \mathbf{B}) \cdot (\mathbf{C} \otimes \mathbf{D}). \tag{4}$$

Matrix rings/groups. Let $M_n(R)$ denote a matrix ring, i.e., the ring of $n \times n$ matrices with coefficients in a ring R. When $M_n(R)$ is called a matrix group, we consider matrix multiplication as the group operation. By default we assume R is a commutative ring with unity. The rank of a matrix $\mathbf{M} \in M_n(R)$ refers to its R-rank.

Let $\mathsf{GL}(n, R)$ be the group of units in $M_n(R)$, i.e., the group of invertible $n \times n$ matrices with coefficients in R. Let $\mathsf{SL}(n, F)$ be the group of $n \times n$ matrices with determinant 1 over a field F. When $q = p^k$ is a prime power, let $\mathsf{GL}(n, q)$, $\mathsf{SL}(n, q)$ denote the corresponding matrix groups over the finite field $\mathbb{F}_q$.

Cryptographic notions. 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 abbreviations for statistically close and computationally indistinguishable.

Definition 1 (Pseudorandom function [29])**.** A family of deterministic functions $\mathcal{F} = \{F_k : D_\lambda \to R_\lambda\}_{\lambda \in \mathbb{N}}$ is *pseudorandom* if there exists a negligible function $\mathrm{negl}(\cdot)$ for any probabilistic polynomial time adversary Adv, such that

$$\left| \Pr_{k,\mathsf{Adv}}[\mathsf{Adv}^{F_k(\cdot)}(1^\lambda) = 1] - \Pr_{O,\mathsf{Adv}}[\mathsf{Adv}^{O(\cdot)}(1^\lambda) = 1] \right| \leq \mathrm{negl}(\lambda),$$

where $O(\cdot)$ denotes a truly random function.

3 Direct Attacks on Matrix PRFs

In this section we stand from the attacker's point of view to examine what are the basic characteristics that a matrix PRF should (or should not) have. Let $\mathbb{G} = M_w(R)$, $h = c \cdot \ell$. We consider read-c matrix PRFs of the form:

$$F : \{0,1\}^\ell \to R, \quad x \mapsto \mathbf{u}_L \cdot \prod_{i=1}^{h} \mathbf{M}_{i, x_i \bmod \ell} \cdot \mathbf{u}_R \tag{5}$$

where $\mathbf{u}_L, \mathbf{u}_R$ denote the left and right bookend vectors. The seed is given by

$$\mathbf{u}_L, \{\mathbf{M}_{i,b} \in \mathbb{G}\}_{i \in [h], b \in \{0,1\}}, \mathbf{u}_R.$$

3.1 Rank attack

We describe the rank attack which runs in time and space $w^{O(c)}$, where w is the dimension of the $\mathbf{M}$ matrices, c is the number of repetitions of each input bits in the branching program steps. The attack is originated from the zeroizing attack plus tensoring analysis in the obfuscation literature [23, 6, 18].

The main idea of the attack is to form a matrix from the evaluations on different inputs. We argue that the rank of such a matrix is bounded by $w^{O(c)}$, whereas for a truly random function, the matrix is full-rank with high probability.

Algorithm 3.1 (Rank attack)**.** The algorithm proceeds as follows.

1. Let $\rho > w^{2c-1}$. Divide the ℓ input bits into 2 intervals $[\ell] = \mathcal{X} \mid \mathcal{Y}$ such that $|\mathcal{X}|, |\mathcal{Y}| \geq \lceil \log \rho \rceil$.
2. For $1 \leq i, j \leq \rho$, evaluate the function F on ρ^2 different inputs of the form $u^{(i,j)} = x^{(i)} \mid y^{(j)} \in \{0,1\}^{\ell}$. Let $v^{(i,j)} \in R$ be the evaluation result on $u^{(i,j)}$:
$$v^{(i,j)} := F(u^{(i,j)})$$
3. Output the rank of matrix $\mathbf{V} = (v^{(i,j)}) \in R^{\rho \times \rho}$.

3.1.1 Analysis for read-once branching programs

First we analyze the case where $c = 1$, i.e. the function is read-once. For a truly random function, the R-rank of $\mathbf{V}$ is ρ with non-negligible probability.

However, for the function F in Eqn. (5), the R-rank of $\mathbf{V}$ is bounded by w, since

$$\mathbf{V} = \begin{pmatrix} v^{(1,1)} & \dots & v^{(1,\rho)} \\ \dots & \dots & \dots \\ v^{(\rho,1)} & \dots & v^{(\rho,\rho)} \end{pmatrix} = \underbrace{\begin{pmatrix} \mathbf{u}_L \cdot \mathbf{M}_{x^{(1)}} \\ \dots \\ \mathbf{u}_L \cdot \mathbf{M}_{x^{(\rho)}} \end{pmatrix}}_{:=\mathbf{X} \in R^{\rho \times w}} \cdot \underbrace{\begin{pmatrix} \mathbf{M}_{y^{(1)}} \cdot \mathbf{u}_R & \dots & \mathbf{M}_{y^{(\rho)}} \cdot \mathbf{u}_R \end{pmatrix}}_{:=\mathbf{Y} \in R^{w \times \rho}}. \tag{6}$$

Here we abuse the subset product notation at $\mathbf{M}_{y^{(j)}}$ by assuming the index of the string $y^{(j)}$ starts at the $(|\mathcal{X}| + 1)^{th}$ step, for $j \in [\rho]$.

3.1.2 Analysis for matrix PRFs with multiple repetitions

The analysis for read-once width w branching programs simply uses the fact that $\mathbf{M}_{x\|y}$ can be written as an inner product of two vectors of length w which depend only on x and y respectively. Here, we show that for read-c width w branching programs, $\mathbf{M}_{x\|y}$ can be written as an inner product of two vectors of length w^{2c-1}. Note that this was already shown in [23, Section 4.2], but we believe our analysis is simpler and more intuitive.

Flattening matrices. For a matrix $\mathbf{A} = \left(\mathbf{a}_1 \mid \dots \mid \mathbf{a}_m\right) \in \mathbb{R}^{n \times m}$, let $\mathsf{flat}(\mathbf{A}) \in \mathbb{R}^{1 \times nm}$ denote the row vector formed by concatenating the rows of $\mathbf{A}$. As it turns out, we can write

$$\mathbf{a}\mathbf{B}_1\mathbf{B}_2 \dots \mathbf{B}_c = \mathsf{flat}(\mathbf{a}\mathbf{B}_1 \otimes \mathbf{B}_2 \otimes \dots \otimes \mathbf{B}_c)\mathbf{J} \tag{7}$$

where $\mathbf{J}$ is a fixed matrix over $\{0,1\}$ independent of $\mathbf{a}, \mathbf{B}_1, \mathbf{B}_2, \ldots, \mathbf{B}_c$.[1] The intuition for the identity is that each entry in the row vector $\mathbf{a}\mathbf{B}_1 \cdots \mathbf{B}_c$ is a linear combination of terms, each a product of entries in $\mathbf{a}\mathbf{B}_1, \ldots, \mathbf{B}_c$, which appears as an entry in $\mathbf{a}\mathbf{B}_1 \otimes \cdots \otimes \mathbf{B}_c$.

In addition, we also have the identity

$$\mathsf{flat}(\mathbf{A}\mathbf{B}) = \mathsf{flat}(\mathbf{A}) \cdot (\mathbf{I}_n \otimes \mathbf{B}) \tag{8}$$

where n is the height of $\mathbf{A}$.[2]

Decomposing read-many branching programs. Given a read-c branching program of width w, we can write $\mathbf{M}_{x\|y}$ as

$$\begin{aligned}
\mathbf{M}_{x\|y} &= \mathbf{u}_L \mathbf{M}_x^1 \mathbf{N}_y^1 \cdots \mathbf{M}_x^c \mathbf{N}_y^c \mathbf{u}_R \\
&= \mathsf{flat}\big((\mathbf{u}_L \mathbf{M}_x^1 \mathbf{N}_y^1) \otimes \cdots \otimes (\mathbf{M}_x^c \mathbf{N}_y^c \mathbf{u}_R)\big) \cdot \mathbf{J} \quad \text{via (7)} \\
&= \mathsf{flat}\Big(\underbrace{(\mathbf{u}_L \mathbf{M}_x^1 \otimes \cdots \otimes \mathbf{M}_x^c)}_{w^{c-1} \times w^c} \cdot \underbrace{(\mathbf{N}_y^1 \otimes \cdots \otimes \mathbf{N}_y^c \mathbf{u}_R)}_{w^c \times w^{c-1}}\Big) \cdot \mathbf{J} \quad \text{via mixed-product} \\
&= \underbrace{\mathsf{flat}\big((\mathbf{u}_L \mathbf{M}_x^1 \otimes \cdots \otimes \mathbf{M}_x^c)\big)}_{1 \times w^{2c-1}} \cdot \underbrace{(\mathbf{I}_{w^{c-1}} \otimes \mathbf{N}_y^1 \otimes \cdots \otimes \mathbf{N}_y^c \mathbf{u}_R) \cdot \mathbf{J}}_{w^{2c-1} \times 1} \quad \text{via (8)}
\end{aligned}$$

That is, $\mathbf{M}_{x\|y}$ can be written as an inner product of two vectors of length w^{2c-1}. Therefore, the rank of $\mathbf{V}$ is at most w^{2c-1}.

Comparison with [23, 6]. We briefly mention that the previous analysis in [23, 6] works by iterating applying the identity

$$\mathsf{flat}(\mathbf{A} \cdot \mathbf{X} \cdot \mathbf{B}) = \mathsf{flat}(\mathbf{X}) \cdot (\mathbf{A}^\top \otimes \mathbf{B})$$

c times along with the mixed-product property to switch the order of the matrix product. (The papers refer to "vectorization" vec, which is the column analogue of flat.) Our analysis is one-shot and avoids this iterative approach, and also avoids keeping track of matrix transposes.

[1] Here's a concrete example:

$$\begin{pmatrix} a_1 & a_2 \end{pmatrix} \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} = \underbrace{\mathsf{flat}\left(\begin{pmatrix} a_1 & a_2 \end{pmatrix} \otimes \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}\right)}_{= \begin{pmatrix} a_1 b_2 & a_2 b_1 & a_1 b_2 & a_2 b_2 \end{pmatrix}} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

[2] Here's a concrete example:

$$\mathsf{flat}\left(\begin{pmatrix} a_1 \\ a_2 \end{pmatrix} \begin{pmatrix} b_1 & b_2 \end{pmatrix}\right) = \begin{pmatrix} a_1 b_1 & a_1 b_2 & a_2 b_1 & a_2 b_2 \end{pmatrix} = \begin{pmatrix} a_1 & a_2 \end{pmatrix} \begin{pmatrix} b_1 & b_2 & & \\ & & b_1 & b_2 \end{pmatrix}$$

Open Problem. Can we prove the following generalization of the rank attack? Let g be a polynomial of total degree at most d in the variables $x_1, \ldots, x_n, y_1, \ldots, y_n$ over $\mathbb{F}_q$ (or even $\mathbb{Z}$), which computes a function $\{0,1\}^n \times \{0,1\}^n \to \mathbb{F}_q$. Now, pick some arbitrary $X_1, \ldots, X_L, Y_1, \ldots, Y_L \in \{0,1\}^n$, and consider the matrix

$$\mathbf{V} := (g(X_i, Y_j)) \in \mathbb{F}_q^{L \times L}$$

Conjecture:

$$\mathsf{rank}(\mathbf{V}) \leq \max\{L, n^{O(d)}\}$$

If the conjecture is true, then we obtain an attack that works not only for matrix products, but basically any low-degree polynomial.

Here's a potential approach to prove the conjecture (based on the analysis of the rank attack). Write g as a sum of monomials g_k. We can write $\mathbf{V}$ as a sum of matrices $\mathbf{V}_k$ where $\mathbf{V}_k := (g_k(X_i, Y_j))$. Each $\mathbf{V}_k$ can be written as a product of two matrices, which allows us to bound the rank of $\mathbf{V}_k$. Then, use the fact that $\mathsf{rank}(\mathbf{V}) \leq \sum_k \mathsf{rank}(\mathbf{V}_k)$. A related question is, can we use this approach to distinguish g from random low-degree polynomials? A related challenge appears here in [1].

3.2 Implication of the rank attack

We briefly discuss the implication of the rank attack to two relevant proposals (or paradigms) of constructing efficient PRFs [12] and cryptographic hash functions [44, 43]. Both proposals use the group operations over a sequence of group elements as the evaluation functions. The rank attack implies when the underlying group $\mathbb{G}$ admits an efficiently computable homomorphism to a matrix group $M_n(R)$, and when each input bit chooses a constant number of steps in the evaluation, then the resulting function is not a PRF (resp. the resulting hash function cannot be used as a random oracle).

Let us remark that our attack does not refute any explicit claims in those two proposals. It mainly serves as a sanity check for the future proposals of instantiating PRFs (resp. hash functions) following those two paradigms. Let us also remark that the rank attack is preventable by adding an one-way extraction function at the end of the evaluation. But when the PRF (resp. hash function) is used inside other applications, an extraction function that is compatible with the application may not be easy to construct. As an example, when the matrix PRFs are used in safeguarding the branching-program obfuscator like [26, 27], it is not clear how to apply an extraction function that is compatible with the obfuscator.

3.2.1 Efficient PRF based on the conjugacy problem

In the conference on mathematics of cryptography at UCI, 2015, Boneh proposed a simple construction of PRF based on the hardness of conjugacy problem, and suggested to look for suitable non-abelian groups for which the conjugacy problem is hard [12]. If such a group is found, it might lead to a PRF that is as efficient as AES. However, even without worrying about efficiency, it is not clear how to find a group where the decisional conjugacy problem is hard.

Here is a brief explanation of the conjugacy problem and the PRF construction [12]. Let K be a non-abelian group, G be a subset of K, H be a subgroup of K. Given $g \xleftarrow{\$} G$, $z = h \circ g \circ h^{-1}$ where $h \xleftarrow{\$} H$, the search conjugacy problem asks to find h.

The PRF construction relies on the following decision version of the conjugacy problem. Let m be a polynomial. For $h \xleftarrow{\$} H$, g_1, g_2, ..., $g_m \xleftarrow{\$} G^m$. The decisional problem asks to distinguish

$$g_1, h \circ g_1 \circ h^{-1}, ..., g_m, h \circ g_m \circ h^{-1}$$

from $2m$ random elements in G.

Let the input be $x \in \{0,1\}^\ell$, the key be $k = g, \{h_{i,b}\}_{i \in [\ell], b \in \{0,1\}}$. Then the following construction is a PRF assuming the decisional conjugacy problem is hard.

$$F_k(x) := h_{\ell,x_\ell} \circ h_{\ell-1,x_{\ell-1}} \circ ... \circ h_{1,x_1} \circ g \circ h^{-1}_{1,x_1} \circ ... \circ h^{-1}_{\ell,x_\ell}$$

The proof follows the augmented cascade technique of [15].

Note that F only has $2\ell - 1$ steps, with each index in the input repeating for at most 2 times. So if G admits an efficient homomorphism to a matrix group, then the rank attack applies.

Finally, let us remark that there are candidate group for which the search conjugacy problem is plausibly hard, e.g. the braid group [33]. But the decisional conjugacy problem over the braid group is broken exactly using a representation as a matrix group [22].

3.2.2 Cryptographic hash functions based on Cayley graphs

We first recall the hard problems on Cayley graphs and their applications in building cryptographic hash functions [41]. Let $\mathbb{G}$ be a finite non-abelian group, and $S = \{s_0, ..., s_m\}$ be a small generation set. The Cayley graph with respect to $(\mathbb{G}, S)$ is defined as follows: each element $v \in \mathbb{G}$ defines a vertex; there is an edge between two vertices v_i and v_j if $v_i = v_j \circ s$ for some $s \in S$. The factorization problem asks to express an element of the group $\mathbb{G}$ as a "short" product of elements from S. For certain groups and generation sets, the factorization problem is conjectured to be hard.

In 1991, Zémor [44] introduced a cryptographic hash function based on a Cayley graph with respect to the group $\mathbb{G} = \mathsf{SL}(2, \mathbb{F}_p)$ and the set $S = \left\{ s_0 = \begin{pmatrix} 1, & 1 \\ 0, & 1 \end{pmatrix}, s_1 = \begin{pmatrix} 1, & 0 \\ 1, & 1 \end{pmatrix} \right\}$. Let the input of the hash function be $x \in \{0,1\}^\ell$. The evaluation of the hash function is simply

$$H(x) := \prod_{i=1}^{\ell} s_{x_i}.$$

The collision resistance of this function is based on the hardness of the factorization problem.

The factorization problem with respect to the original proposal of Zémor was solved by [43]. Then alternative proposals of the group $\mathbb{G}$ and generation set S have since then been given (see the survey of [41]). Most of the groups in these proposals are still matrix groups.

We observe that since H is read-once, if the underlying group $\mathbb{G}$ is a matrix group, then the rank attack is able to distinguish the hash function from a random oracle.

Finally, let us clarify that the original authors of the Cayley hash function proposals do not claim to achieve the random-oracle like properties, and most of the analyses of the Cayley graph-based hash function focus on its collision resistance (which is directly related to the factorization problem). Still, many applications of cryptographic hash functions require random-oracle like properties (e.g. in the Fiat-Shamir transformation), so we think it is worth to point out that the Cayley graph-based hash function does not achieve those strong properties when instantiated with matrix groups.

4 PRFs from Hard Matrix Problems

In this section, we propose plausibly hard problems related to matrix products, from which we can build a matrix PRF using the Naor-Reingold paradigm. We start from a few simple problems and explain how these problems can be solved efficiently. Then we generalize the attack methodology. Finally, we conclude with the final assumptions which survive our cryptanalytic attempts.

4.1 The initial attempts

First Take and the Determinant Attack. Our first assumption sets $\mathbb{G}$ to be the group $\mathsf{GL}(n, p)$ where we think of n as being the security parameter. Let m be an arbitrarily polynomially large integer. The assumption says that the following two distributions are computationally indistinguishable:

$$(\mathbf{A}_1, ..., \mathbf{A}_m, (\mathbf{A}_1\mathbf{B})^k, ..., (\mathbf{A}_m\mathbf{B})^k) \approx_c (\mathbf{A}_1, ..., \mathbf{A}_m, \mathbf{U}_1, ..., \mathbf{U}_m) \tag{9}$$

where all the matrices are chosen uniformly at random from $\mathsf{GL}(n, p)$.

Let us explain the choice of k. When $k = 1$, the assumption is trivially broken since we can just compute $\mathbf{B}$ on the LHS. When k is a constant, we are still able to break the assumption using a linear algebraic technique detailed in Section 3. So we set k to be as large as the security parameter.

Unfortunately, even with a large k the assumption is broken, since on the LHS we have

$$\mathsf{det}((\mathbf{A}_2\mathbf{B})^k) \cdot \mathsf{det}(\mathbf{A}_1)^k = \mathsf{det}((\mathbf{A}_1\mathbf{B})^k) \cdot \mathsf{det}(\mathbf{A}_2)^k$$

In general, any group homomorphism from $\mathbb{G}$ to an Abelian group $\mathcal{H}$ allows us to carry out this attack.

Second Take and the Order Attack. The easy fix for this is to take the group to be $\mathsf{SL}(n, p)$, the group of n-by-n matrices with determinant 1. It is known that *for several choices of n and p*, $\mathsf{SL}(n, p)$ is simple, namely, it has no normal subgroups. Consequently, it admits no non-trivial group homomorphisms to any Abelian group.

Fact 1 (see, e.g., [32])**.** The following are true about the special linear group $\mathsf{SL}(n, p)$.

1. The projective special linear group $\mathsf{PSL}(n, p)$ defined as the quotient $\mathsf{SL}(n, p)/Z(\mathsf{SL}(n, p))$ is simple for any n and p, except when $n = 2$ and $p = 2, 3$. Here, $Z(G)$ denotes the center of group G, the set of elements in G that commute with any other element of G.
2. For n and p where $\mathsf{gcd}(n, p - 1) = 1$, the center of $\mathsf{SL}(n, p)$ is trivial. Namely, $Z(\mathsf{SL}(n, p)) = \{I_n\}$.
3. As a consequence of (1) and (2) above, for $n \geq 3$ and p such that $\mathsf{gcd}(n, p - 1) = 1$, $\mathsf{SL}(n, p)$ is simple.

In particular, we will pick $p = 2$ and $n \geq 3$ to be a large number.

However, we notice that there is a way to break the assumption simply using the group order.

Fact 2 (see, e.g., [32])**.** The order of $\mathsf{SL}(n, p)$ is easily computable. It is

$$r := |\mathsf{SL}(n, p)| = p^{n(n-1)/2} \cdot (p^n - 1) \cdot (p^{n-1} - 1) \cdot \ldots \cdot (p^2 - 1)$$

Therefore, when k is relatively prime to r, we can compute $\mathbf{A}_1\mathbf{B}$ from $(\mathbf{A}_1\mathbf{B})^k$ as follows: let $s = k^{-1} \bmod r$ and compute $\left((\mathbf{A}_1\mathbf{B})^k\right)^s = \mathbf{A}_1\mathbf{B}$. Consequently, the similar assumption for group $\mathsf{SL}(n,p)$ is also broken easily.

One may hope that the assumption holds for certain subgroup of $\mathbb{G} \subset \mathsf{GL}(n,p)$. To rule out the order attack, however, we should choose either 1) to hide the order of group $\mathbb{G}$ or 2) fix the order of group to have many divisors, but neither is a nontrivial. We instead seek another way as follows.

Summary. From the first two attempts we rule out some choices of the group and parameters. Here is a quick summary.

- k has to be as large as the security parameter λ to avoid the rank attack.
- The determinant attack can be generalized to the case when there is an (efficiently computable) homomorphism f from $\mathbb{G}$ to an abelian group H, since it crucially relies on the fact that $f((\mathbf{A}_2\mathbf{B})^k) \cdot f(\mathbf{A}_1)^k = f((\mathbf{A}_1\mathbf{B})^k) \cdot f(\mathbf{A}_2)^k$ for $f = \det$. To rule out this class of attacks, we fix $\mathbb{G}$ to be non-abelian simple group.
- The order attack heavily relies on the fact that one can cancel out $\mathbf{A}_1$ in the left-end of the product. We thus use multiple $\mathbf{A}$'s to avoid this canceling with non-abelian group.

4.2 The first formal assumption and construction

Let $\mathbb{G}$ be a non-commutative simple group where the group elements can be efficiently represented by matrices (for example, the alternating group A_n for a polynomially large $n \geq 5$). Let k be as large as the security parameter λ. Our assumption is

$$\left(\{\mathbf{A}_{i,b}\}_{i\in[k],b\in\{0,1\}}, \prod_{i=1}^{k}(\mathbf{A}_{i,0}\mathbf{B}), \prod_{i=1}^{k}(\mathbf{A}_{i,1}\mathbf{B})\right) \approx_c \left(\{\mathbf{A}_{i,b}\}_{i\in[k],b\in\{0,1\}}, \mathbf{B}_0, \mathbf{B}_1\right) \tag{10}$$

where the matrices $\{\mathbf{A}_{i,b}\}_{i\in[k],b\in\{0,1\}}$, $\mathbf{B}$, $\mathbf{B}_0$ and $\mathbf{B}_1$ are chosen from $U(\mathbb{G})$.

The PRF Construction. The family of pseudorandom functions is defined iteratively as follows.

Construction 4.1. The construction is parameterized by matrices $\mathbf{A}_{1,0}, \mathbf{A}_{1,1}, \ldots, \mathbf{A}_{k,0}, \mathbf{A}_{k,1}$ sampled uniformly random from $\mathbb{G}$.

$$\mathsf{PRF}^{(i)}(x_1x_2\ldots x_i) = \prod_{j=1}^{k}(\mathbf{A}_{j,x_i} \cdot \mathsf{PRF}^{(i-1)}(x_1x_2\ldots x_{i-1}))$$

$$\mathsf{PRF}^{(0)}(\epsilon) = \mathbf{I}$$

where ϵ is the empty string and $\mathbf{I}$ is the identity matrix.

The proof follows a Naor-Reingold style argument and proceeds by showing, inductively, that $\mathsf{PRF}^{(i-1)}(x_1x_2\ldots x_{i-1})$ is pseudorandom. If we now denote this matrix by $\mathbf{B}$,

$$\left(\mathsf{PRF}^{(i)}(x_1x_2\ldots 0), \mathsf{PRF}^{(i)}(x_1x_2\ldots 1)\right) = \left(\prod_{j=1}^{k}(\mathbf{A}_{j,0} \cdot \mathbf{B}), \prod_{j=1}^{k}(\mathbf{A}_{j,1} \cdot \mathbf{B})\right)$$

which, by Assumption 10, is pseudorandom.

4.3 Another assumption and a synthesizer-based PRF construction

In the second assumption, we still choose $\mathbb{G}$ as a non-commutative simple group where the group elements can be efficiently represented by matrices. Let m_1, m_2 be arbitrarily polynomially large integers, $k = O(\lambda)$. Let $\{\mathbf{A}_{i,1}, ..., \mathbf{A}_{i,k} \leftarrow U(\mathbb{G}^k)\}_{i\in[m_1]}$, $\{\mathbf{B}_{j,1}, ..., \mathbf{B}_{j,k} \leftarrow U(\mathbb{G}^k)\}_{j\in[m_2]}$. Our assumption is

$$\left(\prod_{v=1}^{k}(\mathbf{A}_{i,v}\mathbf{B}_{j,v})\right)_{i\in[m_1],j\in[m_2]} \approx_c \left(\mathbf{U}_{i,j} \leftarrow U(\mathbb{G})\right)_{i\in[m_1],j\in[m_2]} \tag{11}$$

The synthesizer-based PRF construction. To assist the construction of a synthesizer-based matrix PRF from Assumption (11), let us first define the lists of indices used in the induction.

Let $k = O(\lambda)$, $v = \lceil \log k \rceil$. Let $\ell \in \text{poly}(\lambda)$ be the input length of the PRF. Let ϵ denote the empty string. Let $||$ be the symbol of list concatenation. For any list S of length t, let S^L denote the sublist of the $\lfloor t/2 \rceil$ items from the left, let S^R denote the sublist of the $t - \lfloor t/2 \rceil$ items from the right.

Define the initial index list as $S_\epsilon := \{i_1, i_2, ..., i_\ell\}$. Define the "counter" list as $C := \{a_1, ..., a_v\}$. Let $r \in \{0,1\}^* \cup \epsilon$, iteratively define S_{r0} and S_{r1} as:

$$\begin{aligned}&\text{if } S_r \text{ is defined and } |S_r| \geq 4v, \quad S_{r0} := S_r^L||C,\ S_{r1} := S_r^R||C\\ &\text{if } S_r \text{ is defined and } |S_r| < 4v, \quad \perp.\end{aligned}$$

Let $d \in \mathbb{Z}$ be the depth of the induction, i.e., any defined list S_r has $|r| \leq d$. We have $2^d \geq \ell \geq \left(\frac{4-1}{3-1}\right)^d = 1.5^d$. Since $\ell \in \text{poly}(\lambda)$, we have $2^d \in \text{poly}(\lambda)$.

Construction 4.2. The PRF is keyed by $2^{4v}\cdot 2^d \in \text{poly}(\lambda)$ random matrices $\{\mathbf{A}_{i,S_r} \leftarrow U(\mathbb{G})\}_{i\in\{0,1\}^{4v}, r\in\{0,1\}^d}$. The evaluation formula $\mathsf{PRF}(x) := \mathsf{PRF}_{S_\epsilon}(x_1x_2\dots x_\ell)$ is defined inductively as

$$\begin{aligned}&\text{if } |S_r| \geq 4v \quad \mathsf{PRF}_{S_r}(x_1x_2\dots x_t) = \prod_{j=1}^{k}\left(\mathsf{PRF}_{S_{r0}}(x_1x_2\dots x_{\lfloor t/2\rceil}\tilde{j}) \cdot \mathsf{PRF}_{S_{r1}}(x_{\lfloor t/2\rceil+1}\dots x_t\tilde{j})\right)\\ &\text{if } |S_r| < 4v \quad \mathsf{PRF}_{S_r}(x_1x_2\dots x_t) = \mathbf{A}_{x_1x_2\dots x_t,S_r}.\end{aligned}$$

where $\tilde{j}$ denotes the bit-decomposition of j.

4.4 Open problems

Open problem 1: Matrix PRF with polynomially many steps. In both of our PRF constructions, the numbers of steps in the final branching program (i.e., the number of matrices in each product) are super-polynomial. In Construction 4.1 it takes roughly $O(k^\ell)$ steps; in Construction 4.2 it takes roughly $O(k^d)$ steps. Although those PRFs are efficiently computable (the key is to reuse intermediate products), the numbers of steps are enormous. Is there a way to obtain a matrix PRF with polynomial number of steps from inductive assumptions?

Let us remark that we do not know any attacks on the simple subset product-based matrix PRF when the matrices are sampled from a non-commutative simple group $\mathbb{G}$, and each input is

repeated for at least λ times. More precisely, sample $\{\mathbf{M}_{i+j\cdot\ell,b} \leftarrow U(\mathbb{G})\}_{i\in[\ell],j\in\{0,\ldots,\lambda-1\},b\in\{0,1\}}$ as the key. Define

$$F : \{0,1\}^\ell \to \mathbb{G}, \quad x \mapsto \prod_{j=0}^{\lambda-1}\left(\prod_{i=1}^{\ell}\mathbf{M}_{i+j\cdot\ell,x_i}\right) \tag{12}$$

We conjecture that F is a PRF, but we do not know how to base its pseudorandomness on a simple assumption.

Open problem 2: Universal matrix PRF. Any PRF in NC^1 gives rise a matrix PRF, with a possibly different order of products and different distributions in the matrices. Is there a *canonical order* and a *canonical matrix distribution* such that the security of any NC^1 PRF can be reduced to one construction? This would possibly give us a universal matrix PRF.

5 Matrix Attacks for the Candidate Block-Local PRG from BBKK18

A pseudorandom generator $f : \{0,1\}^{bn} \to \{0,1\}^m$ is called ℓ-block-local if the input can be separated into n blocks, each of size b bits, such that every output bit of f depends on at most ℓ blocks. When roughly $m \geq \tilde{\Omega}(n^{\ell/2})$[3], there is a generic attack on ℓ-block-local PRGs [8]. Specific to 3-block-local PRGs, no generic attack is known for $m < n^{1.5}$.

In [8], the authors propose a simple candidate ℓ-block-local PRG from group theory, where m can be as large as $n^{\ell/2-\epsilon}$. Let us recall their candidate, with $\ell = 3$ for the simplicity of description. Let $\mathbb{G}$ be a finite group that does not have any abelian quotient group. Choose $3m$ random indices $\left\{i_{j,k} \xleftarrow{\$} [n]\right\}_{j\in[m],k\in[3]}$. The 3-block-local-PRG f is mapping from $\mathbb{G}^n$ to $\mathbb{G}^m$ as

$$f_j(x_1,\ldots,x_n) = x_{j,1} \circ x_{j,2} \circ x_{j,3}.$$

In particular, the authors mentioned that $\mathbb{G}$ can be a non-commutative simple group.

We show that when $\mathbb{G}$ admits an efficiently computable homomorphism to a matrix group $M_w(R)$ (e.g. when $\mathbb{G}$ is an alternating groups A_w with $w \geq 5$), then there is an attack that rules out certain choices of combinations of indices in f. In particular, we show that when $\mathbb{G}$ is chosen as the alternating group, then a non-negligible fraction of the candidates (where the randomness is taken over the choices of the indices) are not PRGs.

The attack uses the fact that for any two matrices $\mathbf{A}, \mathbf{B} \in R^{w\times w}$, $\chi(\mathbf{AB}) = \chi(\mathbf{BA})$, where χ denotes the characteristic polynomial. For simplicity let us assume the group $\mathbb{G}$ is super-polynomially large (e.g. $\mathbb{G} = A_w$ where $w = O(\lambda)$). The distinguisher trys to find four output bits whose indices are of the pattern

$$(a,b,c),(d,e,f),(b,c,d),(e,f,a) \tag{13}$$

where the same letter denote the same index.

Then for these four output group elements represented by matrices $\mathbf{M}_1$, $\mathbf{M}_2$, $\mathbf{M}_3$, $\mathbf{M}_4$, we always have $\chi(\mathbf{M}_1\mathbf{M}_2) = \chi(\mathbf{M}_3\mathbf{M}_4)$ in the real case. In the random case, since we assume $\mathbb{G}$ is super-polynomially large, the characteristic polynomials are unlikely to be equal.

[3] More precisely, $m = \Omega(2^{\ell b})(n+2\ell b)^{\lceil \ell/2 \rceil}$ for the size of each block b.

Now we bound the probability for the existence of Pattern (13) if the indices are chosen randomly. The total number N of different layouts of the indices is:

$$N = n^{3m}$$

The total number M of different layouts of the indices such that Pattern (13) occurs can be lower bounded by fixing Pattern (13) over 4 output bits, and choose the rest arbitrarily. I.e.

$$M \geq n^{3(m-4)}$$

So $M/N \geq n^{-12}$, which means as long as $m \geq 4$, a non-negligible fraction of all the candidate 3-block-local-PRGs can be attacked when instantiated with $\mathbb{G}$ as a matrix group.

The attack can be generalized to smaller $\mathbb{G}$, and larger ℓ. On the positive side, the attack also seem to be avoidable by not choosing the indices that form Pattern (13).

6 Candidate Indistinguishability Obfuscation

In this section we give a candidate construction of indistinguishability obfuscation $\mathcal{O}$, following [18, 27, 11].

Preliminaries. A branching program Γ is a set

$$\Gamma = \left\{ \mathbf{u}_L^{\mathbf{P}} \in \{0,1\}^{1\times w}, \left\{ \mathbf{P}_{i,b} \in \{0,1\}^{w\times w} \right\}_{i\in[h],b\in\{0,1\}}, \mathbf{u}_R, \varpi : \{0,1\}^\ell \to \{0,1\}^h \right\}$$

where w is called width of branching program and ϖ an input-to-index function. We write

$$\Gamma(\mathbf{x}') := \begin{cases} \mathbf{u}_L \mathbf{P}_{\mathbf{x}'} \mathbf{u}_R & \text{if } \mathbf{x}' \in \{0,1\}^h \\ \mathbf{u}_L \mathbf{P}_{\mathbf{x}'} & \text{if } \mathbf{x}' \in \{0,1\}^{<h} \end{cases}$$

We say that a branching program Γ computes a function $f : \{0,1\}^\ell \to \{0,1\}$ if

$$\forall \mathbf{x} \in \{0,1\}^\ell : \Gamma(\varpi(\mathbf{x})) = 0 \iff f(\mathbf{x}) = 1$$

We particularly consider a simple input-to-index function $\varpi : \{0,1\}^\ell \to \{0,1\}^h$ that outputs h/ℓ copies of $\mathbf{x}$, i.e. $\varpi(\mathbf{x}) = \mathbf{x}|\mathbf{x}|\cdots|\mathbf{x}$. We denote $c := h/\ell$ and call this branching program c-input-repeating. We define an index-to-input function $\iota : [h] \to [\ell]$ so that $\iota : x \mapsto (x \bmod \ell) + 1$. For a string $\mathbf{x} \in \{0,1\}^*$, we denote the length of $\mathbf{x}$ by $|\mathbf{x}|$. We say $\mathbf{x}' \in \varpi(\{0,1\}^\ell)$ input-consistent or simply consistent.

Lattice Basics. We briefly describe the basic facts in the lattice problems and trapdoor functions. For more detailed discussion and review we refer [18] to readers. What we need for the construction is, roughly speaking, that there is an algorithm, given matrices $\mathbf{A}$ and $\mathbf{B}$ and a trapdoor $\tau_{\mathbf{A}}$, to sample a (random) matrix $\mathbf{D}$ whose entries follow the discrete Gaussian distribution with small variance such that $\mathbf{AD} = \mathbf{B} \bmod q$. We denote this random small-norm Gaussian $\mathbf{D}$ by $\mathbf{A}^{-1}(\mathbf{B})$ following [18]. Readers who are not interested in the details may skip the detailed definitions and lemmas described here, since they are only used for technical details such as set parameters, etc.

We denote the discrete Gaussian distribution over $\mathbb{Z}^n$ with parameter σ by $D_{\mathbb{Z}^n,\sigma}$. Given matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, the kernel lattice of $\mathbf{A}$ is denoted by

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

Given $\mathbf{y} \in \mathbb{Z}_q^n$ and $\sigma > 0$, 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 $\mathbf{Ad} = \mathbf{y} \bmod q$. We sometimes omit σ when the context is clear.

Definition 2 (Decisional learning with errors (LWE) [42])**.** For $n, m \in \mathbb{N}$ and modulus $q \geq 2$, distributions for secret vector, public matrices, and error vectors $\theta, \pi\chi \subset \mathbb{Z}_q$. An LWE sample w.r.t. these parameters is obtained by sampling $\mathbf{s} \leftarrow \theta^n$, $\mathbf{A} \leftarrow \pi^{n\times m}$, $\mathbf{e} \leftarrow \chi^m$ and outputting $(\mathbf{A}, \mathbf{s}^T\mathbf{A} + \mathbf{e}^T \bmod q)$.

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 6.1 (Standard form [42, 38, 16, 39])**.** *For $n \in \mathbb{N}$ and for any $m = \mathrm{poly}(n)$, $q \leq 2^{\mathrm{poly}(n)}$. Let $\theta = \pi = U(\mathbb{Z}_q)$ and $\chi = D_{\mathbb{Z},\sigma}$ where $\sigma \geq 2\sqrt{n}$. If there exist an efficient (possibly quantum) algorithm that solves $\mathsf{LWE}_{n,m,q,\theta,\pi,\chi}$, then there exists an efficient (possibly quantum) algorithm for approximating SIVP and GapSVP in ℓ_2 norm, in the worst case, within $\tilde{O}(nq/\sigma)$ factors.*

Lemma 6.2 (LWE with small public matrices [14])**.** *If n, m, q, σ are chosen as Lemma 6.1, then $\mathsf{LWE}_{n',m,q,U(\mathbb{Z}_q),D_{\mathbb{Z},\sigma},D_{\mathbb{Z},\sigma}}$ is as hard as $\mathsf{LWE}_{n,m,q,\theta,\pi,\chi}$ for $n' \geq 2n\log q$.*

Lemma 6.3 ([3, 4, 36, 28])**.** *There is a p.p.t. algorithms $\mathsf{TrapSamp}(1^n, 1^m, q)$ that, given modulus $q \geq 2$ and dimension m, n such that $m \geq 2n\log q$, outputs $\mathbf{A} \approx_s U(\mathbb{Z}_q^{n\times m})$ with a trapdoor τ. Further, if $\sigma \geq 2\sqrt{n\log q}$, there is a p.p.t. algorithm that, given $(\mathbf{A},\tau) \leftarrow \mathsf{TrapSam}(1^n, 1^m, q)$ and $\mathbf{y} \in \mathbb{Z}_q^n$, outputs a sample from $\mathbf{A}^{-1}(\mathbf{y},\sigma)$. Further, it holds that*

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

6.1 Construction

Input. The obfuscation algorithm takes as input a c-input-repeating branching program $\Gamma = \{\mathbf{u}_L \in \{0,1\}^{1\times w}, \{\mathbf{P}_{i,b} \in \{0,1\}^{w\times w}\}_{i\in[h],b\in\{0,1\}}, \mathbf{u}_R\}$ computing a function $f : \{0,1\}^{\ell} \to \{0,1\}$.

We modify Γ to a new functionally equivalent branching program Γ' so that it satisfies $\Gamma'(\mathbf{x}') \neq 0$ for all $\mathbf{x}' \notin \varpi(\{0,1\}^h)$ (as well as $\mathbf{x}' \in \{0,1\}^{<h}$). This can be done by padding an input-consistency check program in the right-bottom diagonal of $\mathbf{P}$, which only slightly increases w and the bound of entries. Concretely we follow Construction 6.1.2. For brevity, we just assume that the input program is of the form

$$\Gamma = \{\mathbf{u}_L \in \{0,1,\cdots,T\}^{1\times w}, \{\mathbf{P}_{i,b} \in \{0,1,\cdots,T\}^{w\times w}\}_{i\in[h],b\in\{0,1\}}, \mathbf{u}_R\}$$

and assume that it satisfies the condition above without loss of generality. In particular, $|\Gamma(\varpi(\mathbf{x}))| \leq T$ in this construction.

Obfuscation Procedure.

- Set parameters $n, m, q, \tau, \nu, B \in \mathbb{N}$ and $\sigma \in \mathbb{R}^+$ as in Parameter 6.1.1. Let $d := wn + 5\tau + 3\ell$ be a dimension of pre-encoding.
- Sample a matrix PRF $\{\mathbf{u}_L^{\mathbf{M}} \in \{0,1\}^{1\times 5\tau}, \{\mathbf{M}_{i,b} \in \{0,1\}^{5\tau\times 5\tau}\}_{i\in[h],b\in\{0,1\}}, \mathbf{u}_R^{\mathbf{M}} \in \mathbb{Z}^{5\tau\times 1}\}$ with input length ℓ and c-repetition whose range is $[0, 2^\tau - 1]$. Concretely, we follow Construction 6.1.2. By padding the programs, we may assume that the input program and the matrix PRF share the same input-to-index function $\varpi : \{0,1\}^h \to \{0,1\}^\ell$.
- Sample $\left\{\mathbf{S}_{i,b} \leftarrow D_{\mathbb{Z},\sigma}^{n\times n}\right\}_{i\in[h],b\in\{0,1\}}$ and $\mathbf{a}_h \leftarrow U(\mathbb{Z}_q^{n\times 1})$, and compute pre-encodings as follows:

$$\mathbf{J} := \left(\mathbf{u}_L \otimes \mathbf{1}^{1\times n} || \mathbf{u}_L^{\mathbf{M}}\right), \quad \mathbf{L} := \begin{pmatrix} \mathbf{u}_R \otimes \mathbf{a}_h \\ \mathbf{u}_R^{\mathbf{M}} \end{pmatrix},$$
$$\hat{\mathbf{S}}_{i,b} := \begin{pmatrix} \mathbf{P}_{i,b} \otimes \mathbf{S}_{i,b} & \\ & \mathbf{M}_{i,b} \end{pmatrix} \quad \text{for } i \in [h]$$

  For brevity we write $\mathbf{S}(\mathbf{x}') := \mathbf{1}^{1\times n} \cdot \mathbf{S}_{\mathbf{x}'} \cdot \mathbf{a}_h$. In particular, for all $\mathbf{x}' \in \{0,1\}^h$,

$$\begin{aligned} &\mathbf{J} \cdot \hat{\mathbf{S}}_{\mathbf{x}'} \cdot \mathbf{L} \\ &= \Gamma(\mathbf{x}') \cdot \mathbf{S}(\mathbf{x}') + \mathbf{u}_L^{\mathbf{M}} \cdot \mathbf{M}_{\mathbf{x}'} \cdot \mathbf{u}_R^{\mathbf{M}} \\ &= \begin{cases} \mathsf{PRF}_{\mathbf{M}}(\mathbf{x}) & \text{if } \mathbf{x}' = \varpi(\mathbf{x}) \text{ and } f(\mathbf{x}) = 1 \\ (\neq 0) \cdot \mathbf{S}(\mathbf{x}') + \mathbf{u}_L^{\mathbf{M}} \mathbf{M}_{\mathbf{x}'} \mathbf{u}_R^{\mathbf{M}} & \text{otherwise} \end{cases} \end{aligned}$$

  Note that $\Gamma(\mathbf{x}')$ is a scalar, thus $\otimes$ is just a multiplication.
- Sample error matrices $\mathbf{E}_{i,b}$ from $D_{\mathbb{Z},\sigma}$ with the corresponding dimension and computes

$$\begin{aligned} &\mathbf{A}_J = \mathbf{J} \cdot \mathbf{A}_0 \in \mathbb{Z}^{1\times m} \\ &\mathbf{D}_{i,b} \leftarrow \mathbf{A}_{i-1}^{-1}\left(\hat{\mathbf{S}}_{i,b} \cdot \mathbf{A}_i + \mathbf{E}_{i,b}\right) \in \mathbb{Z}^{m\times m}, i = 1, 2, \cdots, h-1 \\ &\mathbf{D}_{h,b} \leftarrow \mathbf{A}_{h-1}^{-1}\left(\hat{\mathbf{S}}_{h,b} \cdot \mathbf{L} + \mathbf{E}_{h,b}\right) \in \mathbb{Z}^{m\times 1} \end{aligned}$$

Output. The obfuscation algorithms outputs $\{\mathbf{A}_J, \{\mathbf{D}_{i,b}\}_{i\in[h],b\in\{0,1\}}\}$ as an obfuscated program.

Evaluation. For input $\mathbf{x} \in \{0,1\}^\ell$, returns 1 if $|\mathbf{A}_J \cdot \mathbf{D}_{\varpi(\mathbf{x})} \bmod q| < B$, and 0 otherwise.

Correctness. For $\mathbf{x} \in \{0,1\}^{\le h}$ with length h',

$$\mathbf{A}_J \cdot \mathbf{D}_{\mathbf{x}'} = \mathbf{J} \cdot \hat{\mathbf{S}}_{\mathbf{x}'} \cdot \mathbf{A}_{h'} + \mathbf{J} \cdot \sum_{j=1}^{h'} \left(\left(\prod_{i=1}^{j-1} \hat{\mathbf{S}}_{i,x_i} \right) \cdot \mathbf{E}_{j,x_j} \cdot \prod_{k=j+1}^{h'} \mathbf{D}_{k,x_k} \right) \bmod q \tag{14}$$

where $\mathbf{A}_h := \mathbf{L}$. Note that all entries following the discrete Gaussian distribution is bounded by $\sqrt{m}\sigma$ with overwhelming probability. The latter term, GGH15 errors, can be bounded, with all but negligible probability, as follows:

$$\left\| \mathbf{J} \cdot \sum_{j=1}^{h'} \left(\left(\prod_{i=1}^{j-1} \hat{\mathbf{S}}_{i,x_i} \right) \cdot \mathbf{E}_{j,x_j} \cdot \prod_{k=j+1}^{h'} \mathbf{D}_{k,x_k} \right) \right\|_\infty \leq (2wd) \cdot h' \cdot (m\sqrt{m}\sigma \cdot wT)^{h'}$$

In particular, for $\mathbf{x}' = \varpi(\mathbf{x})$ and $f(\mathbf{x}) = 1$, the first term is $\mathsf{PRF}_{\mathbf{M}}(\mathbf{x})$, which is bounded by $2^\tau - 1$. We set $B \geq 2^\tau + (2wd) \cdot h \cdot (m\sqrt{m}\sigma \cdot wT)^h$ so that for every $\mathbf{x}$ satisfying $f(\mathbf{x}) = 1$ the obfuscation outputs correctly.

We also note that, if we set $q > B \cdot \omega(\mathrm{poly}(\lambda))$,

$$\Gamma(\mathbf{x}') = \mathbf{0} \iff \mathbf{x}' = \varpi(\mathbf{x}) \wedge f(\mathbf{x}) = 1$$

holds for any $\mathbf{x}' \in \{0,1\}^{\leq h}$ since we pad the input-consistency check program at the beginning. This implies that the random matrix $\mathbf{A}'_h$ the (partial) evaluation $\mathbf{A}_J \cdot \mathbf{D}_{\mathbf{x}'}$ is not canceled. That is, the probability that the evaluation of obfuscation outputs 1 is negligible for an incomplete, inconsistent input $\mathbf{x}'$ or an input $\mathbf{x}' = \varpi(\mathbf{x})$ satisfying $f(\mathbf{x}) = 0$.

6.1.1 Parameters.

Our parameter settings follow [18, 11], which matches to the current existing safety mechanisms. Let λ be a security parameter of construction and $\lambda_{\mathsf{LWE}} = \mathrm{poly}(\lambda)$ a security parameter of underlying LWE problem. Let $d := wn + 5\tau$ be a dimension of pre-encodings. For trapdoor functionalities, $m = \Omega(d \log q)$ and $\sigma = \Omega(\sqrt{z \log q})$ by Lemma 6.3. Set $n = \Omega(\lambda_{\mathsf{LWE}} \log q)$ and $\sigma = \Omega(\sqrt{\lambda_{\mathsf{LWE}}})$ for the security of LWE as in Lemma 6.1 and 6.2. Set $q \leq (\sigma/\lambda_{\mathsf{LWE}}) \cdot 2^{\lambda_{\mathsf{LWE}}^{1-\epsilon}}$ for an $\epsilon \in (0,1)$. Also for the security proof in our model, we set $2^\tau \geq (2wd) \cdot h \cdot (m\sqrt{m}\sigma \cdot wT)^h \cdot \omega(\mathrm{poly}(\lambda))$. On the other hand, we set $B \geq 2^\tau + (2wd) \cdot h \cdot (m\sqrt{m}\sigma \cdot wT)^h$ and $q \geq B \cdot \omega(\mathrm{poly}(\lambda))$ for the correctness.[4]

6.1.2 Construction of subprograms.

Input-consistency Check Program. We describe a read-once branching program for checking whether $\mathbf{x}' \in \varpi(\{0,1\}^\ell)$; this plays the role of so-called "bundling scalars" or "bundling matrices" in prior constructions. For $i \in [h]$ and $b \in \{0,1\}$, compute $\mathbf{C}_{i,b} \in \mathbb{Z}^{3\ell \times 3\ell}$ as the $\mathsf{diag}(\mathbf{C}_{i,b}^{(1)}, \cdots, \mathbf{C}_{i,b}^{(\ell)})$ where

$$\mathbf{C}_{i,b}^{(k)} = \begin{cases} \mathbf{I}^{3\times 3} & \text{if } \iota(i) \neq k \\ \mathsf{diag}(1,0,1) & \text{if } \iota(i) = k \text{ and } i \leq (c-1)\ell \\ \mathsf{diag}(0,1,1) & \text{if } \iota(i) = k \text{ and } i > (c-1)\ell \end{cases}$$

Let $\mathbf{u}_L^{\mathbf{C}} = B \cdot \mathbf{1}^{1\times 3\ell}$ and $\mathbf{u}_R^{\mathbf{C}} = (1,1,-1)^T \otimes \mathbf{1}^{\ell\times 1}$, where T is an integer satisfying $\|\mathbf{P}(\mathbf{x}')\|_\infty < T$ for all $\mathbf{x}' \in \{0,1\}^{\leq h}$.

Then $\{\mathbf{u}_L^{\mathbf{C}}, \{\mathbf{C}_{i,b}\}_{i\in[h]b\in\{0,1\}}, \mathbf{u}_R^{\mathbf{C}}\}$ is an input-consistency check program, and further $\mathbf{C}(\mathbf{x}') + \mathbf{P}(\mathbf{x}') \neq \mathbf{0}$ for all $\mathbf{x}' \notin \varpi(\{0,1\}^\ell)$ and $\mathbf{x}' \in \{0,1\}^{<h}$. That is, we concretely consider

$$\Gamma' = \left\{ \mathbf{u}'_L = (\mathbf{u}_L \| \mathbf{u}_L^{\mathbf{C}}), \left\{ \mathbf{P}'_{i,b} = \mathsf{diag}(\mathbf{P}_{i,b}, \mathbf{C}_{i,b}) \right\}_{i\in[h],b\in\{0,1\}}, \mathbf{u}'_R = \begin{pmatrix} \mathbf{u}_R \\ \mathbf{u}_R^{\mathbf{C}} \end{pmatrix} \right\}.$$

[4] Note that by adjusting λ_{LWE} appropriately large, all constraint can be satisfied as in [11, Section 4.3]

In particular, this gives $w_{\mathsf{new}} = w+3\ell$ and the bound of entry $T = 2w$. Also we note that $\Gamma'(\varpi(\mathbf{x})) = \Gamma(\varpi(\mathbf{x}))$, thus this is bounded by T.

Remark 6.4. Usual construction of branching programs have a property that $\mathbf{u}_L \cdot \mathbf{P}'_{\mathbf{x}} \in \{0,1\}^{1\times w}$ for all $\mathbf{x}' \in \{0,1\}^{<h}$ and $|\mathbf{P}(\mathbf{x}')| \leq w$, thus we can set $T := 2w$; or set $T = w^h$ safely. In our parameter setting, we used $T = 2w$.

Matrix PRFs. For concreteness we provide the construction of matrix PRFs used in the obfuscation given in [27, Section 4.2]. By Barrington's theorem [10], we know that there exist matrix PRFs that output a random binary value. WLOG, we assume that it is c-input-repetition branching program. We write this as $\{\mathbf{u}_L^{(j)}, \{\mathbf{M}_{i,b}^{(j)}\}_{i\in[h],b\in\{0,1\}}, \mathbf{u}_R^{(j)}\}_{j\in[\tau]}$ that are independent to each others. Note that all entries are binary. We concatenate them as

$$\mathbf{u}_L^{\mathbf{M}} = (\mathbf{u}_L^{(1)} \| \cdots \| \mathbf{u}_L^{(\tau)}), \quad \mathbf{M}_{i,b} = \mathsf{diag}(\mathbf{M}_{i,b}^{(1)}, \cdots, \mathbf{M}_{i,b}^{(\tau)}), \quad \mathbf{v}_R^{\mathbf{M}} = \begin{pmatrix} \mathbf{v}_R^{(1)} \\ 2 \cdot \mathbf{v}_R^{(2)} \\ \cdots \\ 2^{\tau-1} \cdot \mathbf{v}_R^{(\tau)} \end{pmatrix}$$

then $\mathsf{PRF}_{\mathbf{M}} : \mathbf{x} \mapsto \mathbf{u}_L^{\mathbf{M}} \cdot \mathbf{M}_{\varpi(\mathbf{x})} \cdot \mathbf{u}_R^{\mathbf{M}} \in [0, 2^\tau - 1]$ is a pseudorandom function, which is the desired construction. Note that the width of this program is 5τ.

6.2 Security

Security model. We note that almost all known attacks including the recently reported *statistical zeroizing attack* [19], *rank attack* and *subtraction attack* [18] only exploit the evaluations of $\mathbf{x}' \in \varpi(\{0,1\}^\ell)$. While some attacks called *mixed-input attack* are considered in the literature (e.g. [26]), however, there is only one actual attack [17] in such class for GGH15-based obfuscation so far, which only exploits several input-consistent evaluations as well in the first phase to extract the information to run mixed-input attack. Some attack that indeed use the mixed-inputs for other multilinear maps [25, 24], but the first step either uses the valid inputs [40] or decodes the multilinear map using known weakness of the NTRU problem [20].

From this motivation, we consider a restricted class of adversary which can gets oracle access to an input-consistent evaluation oracle

$$O_r : \mathbf{x} \mapsto \mathbf{A}_J \mathbf{D}_{\varpi(\mathbf{x})} \bmod q, \ \forall \mathbf{x} \in \{0,1\}^\ell$$

In our model that we call *input-consistent evaluation model* the purpose of adversary is to obtain any meaningful information of the implementation of Γ beyond the input-output behavior. More concretely, we say that the obfuscation procedure is VBB-secure in the input-consistent evaluation model if any p.p.t. adversary cannot distinguish the oracle O_r from the following oracle

$$F_r(\mathbf{x}) = \begin{cases} U([0, 2^\tau - 1]) & \text{if } f(\mathbf{x}) = 0 \\ U(\mathbb{Z}_q) & \text{otherwise} \end{cases} \tag{15}$$

with non-negligible probability, i.e. $O_r(\cdot) \approx_c F_r(\cdot)$.

Theorem 6.5. *The obfuscation construction $\mathcal{O}$ is VBB-secure in the input-consistent evaluation models.*

The main strategy is to hide the lower bits by embedded matrix PRFs, and hide the higher bits using lattice-based PRFs [7, 14] stated as follows.

Lemma 6.6 ([18, Lemma 7.4]). *Let $h, n, q, b \in \mathbb{N}$ and $\sigma, \sigma^* \in \mathbb{R}$ s.t. $n = \Omega(\lambda \log q)$, $\sigma = \Omega(\sqrt{\lambda \log q})$, $b \geq h \cdot (\sqrt{n}\sigma)^h$, $\sigma^* > \omega(\mathrm{poly}(\lambda)) \cdot b$, $q \geq \sigma^* \omega(\mathrm{poly}(\lambda))$. Define a function family $\mathcal{F} = \{f_{\mathbf{a}} : \{0,1\}^h \to \mathbb{Z}_q^n\}$, for which the key generation algorithm samples $\mathbf{a} \leftarrow U(\mathbb{Z}_q^n)$ as the private key, $\left\{\mathbf{S}_{i,b} \leftarrow D_{\mathbb{Z},\sigma}^{n \times n}\right\}$ as the public parameters. The evaluation algorithm takes input $\mathbf{x}' \in \{0,1\}^h$ and computes*

$$f_{\mathbf{a}}(\mathbf{x}') = \left(\prod_{i=1}^{h} \mathbf{S}_{i,x_i}\right) \cdot \mathbf{a} + \mathbf{e}_{\mathbf{x}'} = \mathbf{S}_{\mathbf{x}'} \cdot \mathbf{a} + \mathbf{e}_{\mathbf{x}'} (\mathrm{mod} q)$$

where $\mathbf{e}_{\mathbf{x}'} \leftarrow D_{\mathbb{Z},\sigma^*}^n$ is freshly sampled for every $\mathbf{x}' \in \{0,1\}^h$. Then, for $d = \mathrm{poly}(\lambda)$, the distribution of evaluations $\{f_{\mathbf{a}}(\mathbf{x}'_1), \cdots, f_{\mathbf{a}}(\mathbf{x}'_d)\}$ over the choice of $\mathbf{a}$ and errors is computationally indistinguishable from d independent uniform random vectors from $\mathbb{Z}_q^n$, assuming the hardness of $\mathsf{LWE}_{n,\mathrm{poly},q,U(\mathbb{Z}_q),D_{\mathbb{Z},\sigma},D_{\mathbb{Z},\sigma}}$.

The proof of the main theorem is as follows.

Proof of Theorem 6.5. We will show that the sequence of $d = \mathrm{poly}(\lambda)$ queries to O_r are indistinguishable to the corresponding queries to F_r as follows.

$$\begin{aligned}
\{O_r(\cdot)\} &= \{\mathbf{x} \mapsto \Gamma(\varpi(\mathbf{x})) \cdot \mathbf{S}(\varpi(\mathbf{x})) + \mathsf{PRF}_{\mathbf{M}}(\varpi(\mathbf{x})) + (\text{GGH15 errors})\}_{k \in [d]} \\
&\approx_c \{\mathbf{x} \mapsto \Gamma(\varpi(\mathbf{x})) \cdot \mathbf{S}(\varpi(\mathbf{x})) + U([0, 2^\tau - 1]) + (\text{GGH15 errors})\}_{k \in [d]} \\
&\approx_s \left\{\mathbf{x} \mapsto \Gamma(\varpi(\mathbf{x})) \cdot (\mathbf{S}(\varpi(\mathbf{x})) + \mathbf{e}_{\varpi(\mathbf{x})}) + U([0, 2^\tau - 1])\right\}_{k \in [d]} \\
&\approx_s \{\mathbf{x} \mapsto \Gamma(\varpi(\mathbf{x})) \cdot U(\mathbb{Z}_q) + U([0, 2^\tau - 1])\}_{k \in [d]} \\
&\approx_s \{F_r(\cdot)\}
\end{aligned}$$

Here, we are using noise-flooding applied to $\Gamma(\varpi(\mathbf{x}))e_{\varpi(\mathbf{x})}+$ (GGH15 errors). More precisely, to invoke Lemma 6.6, it should hold that $2^\tau \geq h \cdot (\sqrt{n}\sigma)^h \cdot \omega(\mathrm{poly}(\lambda))$ and $2^\tau \geq (2wd) \cdot h \cdot (m\sqrt{m}\sigma \cdot wT)^h \cdot \omega(\mathrm{poly}(\lambda))$ to neglect GGH15 errors. □

Remark 6.7 (weakening PRF requirements)**.** We note that we only use the matrix PRF for noise-flooding, and therefore it suffices to relax pseudorandomness of $F : \{0,1\}^\ell \to [0, 2^\tau - 1]$ to the following: for any efficiently computable B-bounded function $g : \{0,1\}^\ell \to [B, -B]$ where $B \ll 2^\tau$, we have

$$\{\mathbf{x} \mapsto F(\mathbf{x})\} \approx_c \{\mathbf{x} \mapsto F(\mathbf{x}) + g(\mathbf{x})\}$$

where $+$ is computed over $\mathbb{Z}$. A similar relaxation has been considered in the context of weaker pseudorandom generators for building IO [5]. For this notion, one could potentially have candidates where each $\mathbf{M}_{i,b}$ is drawn uniformly at random from a Gaussian distribution but where $\mathbf{v}_R^{\mathbf{M}}$ is the same as in Section 6.1.2.

6.3 Comparison

In this section we compare our model to the previous security model in [11].

First, we briefly review the security model in [11]. This model gives a stronger oracle to the adversary that allows the adversary to query a *polynomial* (or circuit) rather than an input $\mathbf{x}$. More precisely, the adversary chooses a circuit C described by $\left\{\beta_{i,b}^{(k)}\right\}_{i\in[h],b\in\{0,1\},k\in K}$ and queries

$$T = \mathbf{A}_J \sum_{k\in K}\prod_{i=1}^{h}(\beta_{i,0}^{(k)}\mathbf{D}_{i,0} + \beta_{1,1}^{(k)}\mathbf{D}_{i,1}) \bmod q$$

to a zero-testing oracle, and learns the value T only if it is sufficiently small compared to q. We index the zerotesting values obtained by the adversary by u, thus T_u is the adversary's u-th successful zerotesting value. The purpose of adversary is to find any non-trivial algebraic relation between T_u's and pre-encodings $\hat{\mathbf{S}}$.[5] Despite the generality of oracle inputs, the statistical zeroizing attacks in [19] do not fall into this class; the adversary using the statistical zeroizing attacks is to check if an inequality holds.

On the other hand, our model gives an input-consistent oracle to adversary which is much weaker. Instead, the purpose of adversary is to find any information beyond input-output behavior of the program. That is, we do not restrict the goal of adversary to computing a nontrivial algebraic relations. This freedom allows us to capture almost all existing attacks.

An interesting question is to design a model that embrace both models, and construct a secure obfuscation procedure in such model. A candidate model is to allow the adversary to access both oracles described above. Note that [11, Lemma 8] states that the set of adversary's successful zerotest is essentially a set of polynomially-many linear sum of input-consistent evaluations. With this lemma in mind, an obfuscation procedure satisfying the corresponding lemma as well as the VBB security in the input-consistent evaluation model may satisfy a meaningful security in this model.

Acknowledgments.

We would like to thank Jiseung Kim, Alex Lombardi, Takashi Yamakawa and Mark Zhandry for helpful discussions.

References

[1] Scott Aaronson. Arithmetic natural proofs theory is sought, 2008. https://www.scottaaronson.com/blog/?p=336, Accessed: 2018-02-27.

[2] Shweta Agrawal. Indistinguishability obfuscation without multilinear maps: New methods for bootstrapping and instantiation. In *EUROCRYPT, Part I*, pages 191–225, 2019.

[3] 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.

[5] The original model is more general. For example, they considered GGH15 maps over general graphs instead of source-to-sink path, and allows the adversary to query much general polynomials. Still every adversary's query in this model is essentially of the described form.

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

[5] Prabhanjan Ananth, Aayush Jain, Huijia Lin, Christian Matt, and Amit Sahai. iO without multilinear maps: New paradigms via low-degree weak pseudorandom generators and security amplification. In *CRYPTO*, 2019.

[6] 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.

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

[8] Boaz Barak, Zvika Brakerski, Ilan Komargodski, and Pravesh K. Kothari. Limits on low-degree pseudorandom generators (or: Sum-of-squares meets program obfuscation). In *EUROCRYPT (2)*, volume 10821 of *Lecture Notes in Computer Science*, pages 649–679. Springer, 2018.

[9] 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.

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

[11] James Bartusek, Jiaxin Guan, Fermi Ma, and Mark Zhandry. Return of GGH15: provable security against zeroizing attacks. In *Theory of Cryptography - 16th International Conference, TCC 2018, Panaji, India, November 11-14, 2018, Proceedings, Part II*, pages 544–574, 2018.

[12] Dan Boneh. The dan and craig show. https://www.youtube.com/watch?v=m4lv0lXI5uU, 2015. Accessed: 2019-05-17.

[13] Dan Boneh, Yuval Ishai, Alain Passelègue, Amit Sahai, and David J. Wu. Exploring crypto dark matter: - new simple PRF candidates and their applications. In *TCC (2)*, volume 11240 of *Lecture Notes in Computer Science*, pages 699–729. Springer, 2018.

[14] 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.

[15] Dan Boneh, Hart William Montgomery, and Ananth Raghunathan. Algebraic pseudorandom functions with improved efficiency from the augmented cascade. In *ACM Conference on Computer and Communications Security*, pages 131–140, 2010.

[16] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. Classical hardness of learning with errors. In *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 575–584, 2013.

[17] 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.

[18] Yilei Chen, Vinod Vaikuntanathan, and Hoeteck Wee. GGH15 beyond permutation branching programs: Proofs, attacks, and candidates. In *Advances in Cryptology - CRYPTO 2018, Part II*, pages 577–607, 2018.

[19] Jung Hee Cheon, Wonhee Cho, Minki Hhan, Jiseung Kim, and Changmin Lee. Statistical zeroizing attack: Cryptanalysis of candidates of BP obfuscation over GGH15 multilinear map. In *Advances in Cryptology - CRYPTO 2019, Part III*, pages 253–283, 2019.

[20] Jung Hee Cheon, Minki Hhan, Jiseung Kim, and Changmin Lee. Cryptanalyses of branching program obfuscations over GGH13 multilinear map from the NTRU problem. In *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part III*, pages 184–210, 2018.

[21] Jung Hee Cheon, Minki Hhan, Jiseung Kim, and Changmin Lee. Cryptanalysis on the HHSS obfuscation arising from absence of safeguards. *IEEE Access*, 6:40096–40104, 2018.

[22] Jung Hee Cheon and Byungheup Jun. A polynomial time algorithm for the braid diffie-hellman conjugacy problem. In *CRYPTO*, volume 2729 of *Lecture Notes in Computer Science*, pages 212–225. Springer, 2003.

[23] 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.

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

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

[26] 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.

[27] Sanjam Garg, Eric Miles, Pratyay Mukherjee, Amit Sahai, Akshayaram Srinivasan, and Mark Zhandry. Secure obfuscation in a weak multilinear map model. In *Theory of Cryptography Conference, TCC 2016-B,Part II*, pages 241–268, 2016.

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

[29] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986.

[30] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Predicate encryption for circuits from LWE. In *CRYPTO*, pages 503–523, 2015.

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

[32] Kiyoshi Igusa. Notes on the special linear group. http://people.brandeis.edu/~igusa/Math131b/SL.pdf, Accessed: 2018-02-28.

[33] Ki Hyoung Ko, Sangjin Lee, Jung Hee Cheon, Jae Woo Han, Ju-Sung Kang, and Choonsik Park. New public-key cryptosystem using braid groups. In *CRYPTO*, volume 1880 of *Lecture Notes in Computer Science*, pages 166–183. Springer, 2000.

[34] Allison B. Lewko and Brent Waters. Efficient pseudorandom functions from the decisional linear assumption and weaker variants. In *ACM CCS*, pages 112–120, 2009.

[35] Huijia Lin and Stefano Tessaro. Indistinguishability obfuscation from trilinear maps and block-wise local prgs. In *CRYPTO, Part I*, pages 630–660, 2017.

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

[37] Moni Naor and Omer Reingold. Number-theoretic constructions of efficient pseudo-random functions. In *FOCS*, pages 458–467. IEEE Computer Society, 1997.

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

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

[40] Alice Pellet-Mary. Quantum attacks against indistinguishablility obfuscators proved secure in the weak multilinear map model. In *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part III*, pages 153–183, 2018.

[41] Christophe Petit and Jean-Jacques Quisquater. Rubik's for cryptographers. *IACR Cryptology ePrint Archive*, 2011:638, 2011.

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

[43] Jean-Pierre Tillich and Gilles Zémor. Hashing with sl_2. In *CRYPTO*, volume 839 of *Lecture Notes in Computer Science*, pages 40–49. Springer, 1994.

[44] Gilles Zémor. Hash functions and graphs with large girths. In *EUROCRYPT*, volume 547 of *Lecture Notes in Computer Science*, pages 508–511. Springer, 1991.