Cryptanalyses of Branching Program Obfuscations over GGH13 Multilinear Map from the NTRU Problem

Jung Hee Cheon, Minki Hhan, Jiseung Kim, Changmin Lee

Seoul National University, Republic of Korea

Abstract. In this paper, we propose cryptanalyses of all existing indistinguishability obfuscation (*iO*) candidates based on branching programs (BP) over GGH13 multilinear map for all recommended parameter settings. To achieve this, we introduce two novel techniques, *program converting* using NTRU-solver and *matrix zeroizing*, which can be applied to a wide range of obfuscation constructions and BPs compared to previous attacks. We then prove that, for the suggested parameters, the existing general-purpose BP obfuscations over GGH13 do not have the desired security. Especially, the first candidate indistinguishability obfuscation with input-unpartitionable branching programs (FOCS 2013) and the recent BP obfuscation (TCC 2016) are not secure against our attack when they use the GGH13 with recommended parameters. Previously, there has been no known polynomial time attack for these cases.

Our attack shows that the lattice dimension of GGH13 must be set much larger than previous thought in order to maintain security. More precisely, the underlying lattice dimension of GGH13 should be set to $n = \tilde{\Theta}(\kappa^2 \lambda)$ to rule out attacks from the subfield algorithm for NTRU where κ is the multilinearity level and λ the security parameter.

Keywords: Obfuscations, multilinear maps, graded encoding schemes, NTRU.

1 Introduction

Constructing a general-purpose program obfuscation has been a long standing coveted open problem [8, 9] in spite of their fruitful applications. At FOCS 2013, Garg *et al.* suggested the first plausible candidate general-purpose indistinguishability obfuscation (GGHRSW) [23] using branching program (BP) representation of functions [10]. This first candidate of *iO* has ignited the various subsequent studies [3, 5–7, 15, 24, 30, 32, 34] on obfuscations, all of which stand on the cryptographic multilinear maps.

To date, there are three plausible candidates of multilinear map; the first is due to Garg, Gentry, and Halevi [22] (GGH13), the second is due to Coron, Lepoint, and Tibouchi [19] and the last is due to Gentry, Gorbunov, and Halevi [25]. The security of three candidates are not well clarifed, whereas some works [3, 7, 15, 30, 34] claim the security under the idealized model, so-called the *generic multilinear map model*.

Recently several works try to overcome this gap [6,24,29]. In particular, Garg *et al.* proved the security of the slightly modified first candidate *iO* construction (GMMSSZ) under the *weak multilinear map model* of GGH13, which captures all existing polynomial time attacks on BP obfuscations over GGH13 multilinear map [24]. Despite the provable security under these models, the practical security of obfuscations over GGH13 is still in dubious nature.

Direct attack to GGH13. As a direct method of analyzing obfuscations over GGH13, we may consider attacks on the GGH13 encoding scheme. The latent hardness problems of GGH13 are the (overstretched) NTRU problem and the short generator of principal ideal generator problem (SPIP).

The subfield attacks, proposed by Albrecht *et al.* and Cheon *et al.* independently [1,18], are the most notable algorithms to solve the NTRU problem. These attacks shows that the underlying NTRU problem of GGH13-based obfuscation is solved in polynomial time whenever the multilinear level κ is larger than the security parameter λ. By combining this with the algorithms to solve SPIP [12–14,20], GGH13 is broken in classical subexponential time on security parameter λ for the instantiations in [2,27] or quantum polynomial time. This work shows that the parameters of GGH13 should be set to prevent either the algorithms for NTRU or PIP.[1]

Attacks on BP Obfuscations over GGH13. For obfuscations over GGH13 multilinear map, several cryptanalyses have also been suggested. The *annihilation attack* introduced by Miles *et al.* [31] showed that some constructions of single/dual input BP obfuscations [3,6,7,30] do not have the desired security when they are used for general-purpose and implemented with GGH13. The authors presented a very simple example of BPs which are threatened by annihilation attacks. Soon after, Apon *et al.* [4] extended the range of annihilation attacks to BPs generated by Barrington's theorem [10] which is the fundamental method to transform $\mathcal{NC}^1$ circuits into bounded width BPs.

Chen *et al.* [16] presented another attack on BP obfuscation over GGH13 multilinear map. They showed that there exist two functionally equivalent programs with a special property called *input-partitionable*, and their obfuscated programs by GGHRSW can be efficiently distinguished.

Limitations of Previous Works. Despite the diverse attacks on BP obfuscations over GGH13 multilinear map, GGHRSW remains secure against all known PPT attack when it only takes *input-unpartitionable* BPs as input, such as BPs generated by Barrington's theorem. Meanwhile, there is no known polynomial time attack for multi-input branching program obfuscations including GMMSSZ. We also remark that the direct approach [1], with the current best algorithm to solve SPIP [13,20], has the classical exponential running time with respect to security parameter λ when the dimension n of the base number field satisfies $n = \varOmega(\lambda^2)$.

[1] Indeed, the parameters of original paper [22] are already set to be $n = \tilde{\varTheta}(\kappa\lambda^2)$ so that known classical algorithms for PIP require exponential time for λ. On the other hands, the improved parameters [2,27] allow the subexponential time attacks.

Our Contribution. We present distinguishing attacks on candidates BP *iO* over GGH13 multilinear map based on the algorithm to solve the NTRU problem. With the novel two techniques, *program converting* and *matrix zeroizing attack*, we show that existing general-purpose BP obfuscations cannot achieve the desired security when the obfuscations use GGH13 with proposed parameters in [2,22,27]. In other words, there are two functionally equivalent BPs with same length such that their obfuscations obtained by an existing BP obfuscations over GGH13 can be distinguished in polynomial time for the suggested parameters.

Our attack is applicable to wide range of obfuscations and BPs compared to the previous attacks. In particular, we show that multi-input BP obfuscations such as GMMSSZ construction are insecure in the NTRU-solvable parameter regime. Further, we show that the first candidate indistinguishability obfuscation GGHRSW based on GGH13 with current parameters also does not have the desired security even if it only obfuscates input-unpartitionable BPs including branching programs generated by Barrington's theorem. Although a new property of BPs called *linear relationally inequivalence* is exploited in our attack, we show that various pairs of BPs satisfy this property.

As a result, we show that the BP obfuscations based on GGH13 multilinear map with suggested parameters are broken using the algorithm for NTRU solely. Therefore the underlying lattice dimension n of GGH13 should be set to $n = \tilde{\Theta}(\kappa^2\lambda)$ to maintain 2^λ security of obfuscation schemes. This implies the *iO* based on GGH13 is even much inefficient than the previous results [1,28].

1.1 Technical Overview

Here we briefly show how our attack is applied to simplified GGHRSW.

Simplified GGHRSW Obfuscation. Let $P = \{\boldsymbol{M}_{i,b} \in \mathbb{Z}^{d\times d}\}_{b\in\{0,1\},1\le i\le \ell}$ be a set of matrices corresponding to a single input BP such that

$$P(\boldsymbol{x}) := \begin{cases} 0 & \text{if } \prod_{i=1}^{\ell} \boldsymbol{M}_{i,x_i} = \boldsymbol{I}_d \\ 1 & \text{if } \prod_{i=1}^{\ell} \boldsymbol{M}_{i,x_i} \neq \boldsymbol{I}_d, \end{cases}$$

where x_i is the i-th bit of $\boldsymbol{x}$. The obfuscator randomizes the given BP over several steps.

1. Sample random and independent scalars $\{\alpha_{i,b}, \alpha'_{i,b}\}_{b\in\{0,1\},1\le i\le\ell}$ such that $\prod_{i=1}^{\ell} \alpha_{i,x_i} = \prod_{i=1}^{\ell} \alpha'_{i,x_i}$ for all $\boldsymbol{x} \in \{0,1\}^\ell$. [2]
2. Sample bookend vectors $\{\boldsymbol{s}, \boldsymbol{t}, \boldsymbol{s}', \boldsymbol{t}'\}$ such that $\boldsymbol{s}\cdot\boldsymbol{t} = \boldsymbol{s}'\cdot\boldsymbol{t}'$.
3. Sample invertible matrices $\{\boldsymbol{K}_i, \boldsymbol{K}'_i \in \mathbb{Z}^{d\times d}\}_{0\le i\le\ell}$ and set

$$\begin{aligned} \boldsymbol{R}_0 &= \boldsymbol{s}\cdot\boldsymbol{K}_0^{-1}, & \boldsymbol{R}'_0 &= \boldsymbol{s}'\cdot\boldsymbol{K}_0'^{-1} \\ \boldsymbol{R}_{i,b} &= \boldsymbol{\alpha_{i,b}}\cdot\boldsymbol{K}_{i-1}\cdot\boldsymbol{M}_{i,b}\cdot\boldsymbol{K}_i^{-1}, & \boldsymbol{R}'_{i,b} &= \boldsymbol{\alpha}'_{i,b}\cdot\boldsymbol{K}'_{i-1}\cdot\boldsymbol{I}_d\cdot\boldsymbol{K}_i'^{-1} \\ \boldsymbol{R}_{\ell+1} &= \boldsymbol{K}_\ell\cdot\boldsymbol{t}, & \boldsymbol{R}'_{\ell+1} &= \boldsymbol{K}'_\ell\cdot\boldsymbol{t}'. \end{aligned}$$

[2] In fact $\alpha_{i,b} = \alpha'_{i,b}$ should holds in this simplified setting, but we do not use this equality to give the idea of our attack.

For the sake of simplicity, we write $\boldsymbol{R}_{0,b}$, $\boldsymbol{R}_{\ell+1,b}$, $\boldsymbol{R}'_{0,b}$, and $\boldsymbol{R}'_{\ell+1,b}$ to denote $\boldsymbol{R}_0$, $\boldsymbol{R}_{\ell+1}$, $\boldsymbol{R}'_0$, and $\boldsymbol{R}'_{\ell+1}$, respectively. The randomized BP can then maintain the same functionality as the following evaluation, where $x_0, x_{\ell+1}$ are 0.

$$P(\boldsymbol{x}) = \begin{cases} 0 & \text{if } \prod_{i=0}^{\ell+1} \boldsymbol{R}_{i,x_i} - \prod_{i=0}^{\ell+1} \boldsymbol{R}'_{i,x_i} = 0 \\ 1 & \text{if } \prod_{i=0}^{\ell+1} \boldsymbol{R}_{i,x_i} - \prod_{i=0}^{\ell+1} \boldsymbol{R}'_{i,x_i} \neq 0. \end{cases}$$

As a final step, each entry of the $\boldsymbol{R}_i$ and $\boldsymbol{R}'_i$ is encoded through the GGH13 multilinear map. Let $\mathcal{R} = \mathbb{Z}[X]/\langle X^n + 1\rangle$. The plaintext space and encoding space of GGH13 multilinear map is specified by $\mathcal{R}_{\boldsymbol{g}} = \mathcal{R}/\langle \boldsymbol{g}\rangle$ with some small element $\boldsymbol{g} \in \mathcal{R}$ and $\mathcal{R}_q = R/\langle q\rangle$ with some large integer $q \in \mathbb{Z}$, respectively. In GGH13 multilinear map, a random and invertible element $\boldsymbol{z} \in \mathcal{R}_q$ is sampled. Then the encoding of m is of the form $\mathsf{enc}(m) = [(\boldsymbol{r} \cdot \boldsymbol{g} + m)/\boldsymbol{z}]_q$ for some small random element $\boldsymbol{r} \in \mathcal{R}$. The smallness of $\boldsymbol{g}$ and $\boldsymbol{r}$ implies that the size of the numerator is quite smaller than q. We write $\mathsf{enc}(\boldsymbol{R}_{i,b})$ to denote the matrix whose entries are encoding of entries of $R_{i,b}$.

Then, in the case of $P(\boldsymbol{x}) = 0$, evaluation of the encoded BP over input $\boldsymbol{x}$ can be computed as follows:

$$\prod_{i=0}^{\ell+1} \mathsf{enc}(\boldsymbol{R}_{i,x_i}) - \prod_{i=0}^{\ell+1} \mathsf{enc}(\boldsymbol{R}'_{i,x_i}) = \left[\frac{\boldsymbol{e} \cdot \boldsymbol{g}}{\boldsymbol{z}^{\ell+2}}\right]_q$$

where the term $\boldsymbol{e}$ is the small noise element of $\mathcal{R}$. If it is evaluated for another input $\boldsymbol{x}$, the numerator of the evaluated value cannot be a multiple of $\boldsymbol{g}$.

In order to check whether the numerator of the evaluation value of the encoded BP is a zero or not, the GGH13 multilinear map provide a zerotesting parameter $\boldsymbol{p}_{zt} = [(\boldsymbol{h} \cdot \boldsymbol{z}^{\ell+2})/\boldsymbol{g}]_q$ for some element $\boldsymbol{h} \in \mathcal{R}$ of size $\approx \sqrt{q}$. More precisely, when the $\boldsymbol{p}_{zt}$ is multiplied by the evaluated value, it is of the form $\boldsymbol{h} \cdot \boldsymbol{r}'$ and its size is much smaller than q if the numerator is a multiple of $\boldsymbol{g}$. Otherwise it is a large value. Hence, one can publicly test that whether the plaintext of the encoding is zero or not and an encoded BP give the same functionality with the original BP by employing the zerotesting parameter $\boldsymbol{p}_{zt}$.

In summary, the GGHRSW obfuscator outputs the following set as an obfuscated BP.

$$\{\mathsf{enc}(\boldsymbol{R}_{i,b}), \mathsf{enc}(\boldsymbol{R}'_{i,b}), \boldsymbol{p}_{zt}\}$$

Goal of Cryptanalysis on Simplified GGHRSW Obfuscation. The simplified GGHRSW obfuscation given above is called *indistinguishability obfuscation* if the following statement holds: For every two BPs $P^0 = \{\boldsymbol{M}^0_{i,b}\}$, and $P^1 = \{\boldsymbol{M}^1_{i,b}\}$ with the same size and the same functionality and randomly chosen $c \in \{0, 1\}$, any PPT adversary cannot recover c from the given obfuscated program $\{\mathsf{enc}(\boldsymbol{R}^c_{i,b}), \mathsf{enc}(\boldsymbol{R}'^c_{i,b}), \boldsymbol{p}_{zt}\}$.

In other words, our purpose of the cryptanalysis is to recover such c for appropriately given P^0, P^1 and its obfuscation.

Program Converting Technique In the first step, we remove the modulus q using the algorithm for NTRU. The $(1,1)$ and $(1,2)$ components of the $\mathsf{enc}(\boldsymbol{R}_{1,1})$ are of the form $[(\boldsymbol{r}_{1,1} \cdot \boldsymbol{g} + m_{1,1})/\boldsymbol{z}]_q$ and $[(\boldsymbol{r}_{1,2} \cdot \boldsymbol{g} + m_{1,2})/\boldsymbol{z}]_q$, respectively. The ratio $[(\boldsymbol{r}_{1,1} \cdot \boldsymbol{g} + m_{1,1})/(\boldsymbol{r}_{1,2} \cdot \boldsymbol{g} + m_{1,2})]_q$ of two encodings can be understood as an instance of the NTRU problem.

By solving the NTRU problem, we can obtain multiples of the denominator and numerator

$$\boldsymbol{\beta} \cdot (\boldsymbol{r}_{1,1} \cdot \boldsymbol{g} + m_{1,1}, \boldsymbol{r}_{1,2} \cdot \boldsymbol{g} + m_{1,2}) \in \mathcal{R}^2$$

for some small element $\boldsymbol{\beta} \in \mathcal{R}$. Further, dividing $\boldsymbol{\beta} \cdot (\boldsymbol{r}_{1,1} \cdot \boldsymbol{g} + m_{1,1})$ by a $[(\boldsymbol{r}_{1,1} \cdot \boldsymbol{g} + m_{1,1})/\boldsymbol{z}]_q$, we can compute $[\boldsymbol{\beta} \cdot \boldsymbol{z}]_q$. By multiplying this value to all entries of $\mathsf{enc}(\boldsymbol{R}_{i,b})$ and $\mathsf{enc}(\boldsymbol{R}'_{i,b})$, we replace $1/\boldsymbol{z}$ with a small element $\boldsymbol{\beta}$. The obtained entries are of the form $\boldsymbol{\beta} \cdot (\boldsymbol{r}_{j,k} \cdot \boldsymbol{g} + m_{j,k})$, which can be understood as an element defined in $\mathcal{R}$, not $\mathcal{R}_q$ due to its small size. We denote these new BP matrices with entries in $\mathcal{R}$ by $\{\boldsymbol{D}_{i,b}\}$ and $\{\boldsymbol{D}'_{i,b}\}$, respectively.

Next we consider an input $\boldsymbol{x}$ such that $P(\boldsymbol{x}) = 0$.[3] The corresponding computation of matrices $\boldsymbol{R}$ is zero, thus the following equation holds over $\mathcal{R}$ for such input.

$$\prod_{i=0}^{\ell+1} \boldsymbol{D}_{i,x_i} - \prod_{i=0}^{\ell+1} \boldsymbol{D}'_{i,x_i} = \boldsymbol{e} \cdot \boldsymbol{g} \cdot \boldsymbol{\beta}^{\ell+2}$$

Hence, the term is a multiple of $\boldsymbol{g}$. Using the same procedure for other zeros of P, one can recover several multiples of $\boldsymbol{g}$ and then we can recover a basis of ideal $\langle \boldsymbol{g} \rangle$ using lattice algorithms.

Then we can do a plain-like procedure using the above results. More precisely, the following equations hold.

$$\begin{aligned} Eval_{\boldsymbol{D}}(\boldsymbol{x}) &:= \prod_{i=0}^{\ell+1} \boldsymbol{D}_{i,x_i} = \prod_{i=0}^{\ell+1} \alpha_{i,x_i} \cdot \boldsymbol{s} \cdot \prod_{i=1}^{\ell} \boldsymbol{M}^c_{i,x_i} \cdot \boldsymbol{t} \pmod{\boldsymbol{g}} \\ Eval'_{\boldsymbol{D}}(\boldsymbol{x}) &:= \prod_{i=0}^{\ell+1} \boldsymbol{D}'_{i,x_i} = \prod_{i=0}^{\ell+1} \alpha'_{i,x_i} \cdot \boldsymbol{s}' \cdot \prod_{i=1}^{\ell} \boldsymbol{I}_d \cdot \boldsymbol{t}' \pmod{\boldsymbol{g}} \end{aligned}$$

Removing Scalars. In the above step, we removed the modulus q using the solutions of the NTRU problem and obtained matrices $\{\boldsymbol{D}_{i,b}, \boldsymbol{D}'_{i,b}\}$ and a basis of ideal $\langle \boldsymbol{g} \rangle$. We now remove the effects of scalars $\boldsymbol{\alpha}$. $Eval_{\boldsymbol{D}}(\boldsymbol{x})$ and $Eval'_{\boldsymbol{D}}(\boldsymbol{x})$ share the same scalar $\prod_{i=0}^{\ell+1} \alpha_{i,x_i} = \prod_{i=0}^{\ell+1} \alpha'_{i,x_i}$ due to its definition. Thus, we can compute

$$Eval_{\boldsymbol{D}}(\boldsymbol{x})/Eval'_{\boldsymbol{D}}(\boldsymbol{x}) = 1/(\boldsymbol{s}' \cdot \boldsymbol{t}') \cdot \left(\boldsymbol{s} \cdot \prod_{i=1}^{\ell} \boldsymbol{M}^c_{i,x_i} \cdot \boldsymbol{t} \right) \pmod{\boldsymbol{g}}.$$

We note that these values $Eval_{\boldsymbol{D}}(\boldsymbol{x})/Eval'_{\boldsymbol{D}}(\boldsymbol{x})$ all share the same scalar $1/(\boldsymbol{s}' \cdot \boldsymbol{t}') \pmod{\boldsymbol{g}}$.

[3] Because of this step, our attack cannot be applied to BP obfusaction for evasive functions.

Matrix Zeroizing Attack. At last we introduce the *matrix zeroizing attack.* We denote $Eval_{\boldsymbol{M}^0}(\boldsymbol{x})$ and $\widetilde{Eval}_{\boldsymbol{D}}(\boldsymbol{x})$ as $\prod_{i=1}^{\ell} \boldsymbol{M}^0_{i,x_i}$ and $Eval_{\boldsymbol{D}}(\boldsymbol{x})/Eval'_{\boldsymbol{D}}(\boldsymbol{x})$, respectively.

Then, for several $Eval_{\boldsymbol{M}^0}(\boldsymbol{x}_j)$ for $1 \leq j \leq \tau$, we can find a vector $\boldsymbol{q} = (q_1, \cdots, q_\tau)$ such that $\sum_{j=1}^{\tau} q_j \cdot Eval_{\boldsymbol{M}^0}(\boldsymbol{x}_j) = \boldsymbol{0}_d$, where $\boldsymbol{0}_d$ is a zero matrix. If $c = 1$ so that the obfuscated BP is derived from P^0, the following equation also holds.

$$\sum_{j=1}^{\tau} c_j \cdot \widetilde{Eval}_{\boldsymbol{D}}(\boldsymbol{x}_j) = \boldsymbol{0}_d \pmod{\boldsymbol{g}}$$

Otherwise, it would not be zero $\pmod{\boldsymbol{g}}$.

As a result, we can distinguish two obfuscated program efficiently when we know corresponding branching programs. We remark that the matrix zeroizing attack and removing scalars step are slightly different for the other BP obfuscations.

Organization. In Section 2, we introduce the indistinguishability obfuscation, matrix branching program and GGH13 multilinear map. In Section 3, we show main results of our cryptanalyses on BP obfuscations over GGH13 multilinear map. We describe the attackable BP obfuscation Model over GGH13 throughout the Section 4. In addition, we present the algorithm called program converting technique in Section 5. We last propose the matrix zeroizing attack in Section 6.

2 Preliminaries

Notations. The set $\{1, \cdots, n\}$ is denoted by $[n]$ for a positive integer n. The set of integers modulo p is denoted by $\mathbb{Z}_p := \mathbb{Z}/p\mathbb{Z}$. All elements in $\mathbb{Z}_p$ are considered as integers in $(-p/2, p/2]$. We use the bold letters to denote matrices, vectors and elements of ring. For $\boldsymbol{a} = a_0 + \cdots + a_{n-1} \cdot X^{n-1} \in \mathcal{R} = \mathbb{Z}[X]/\langle X^n + 1\rangle$, the size of $\boldsymbol{a}$ means the Euclidean norm of the coefficient vector $(a_0, \cdots, a_{n-1})$. We denote (j,k)-th entry of matrix $\boldsymbol{M}$ by $\boldsymbol{M}[j,k]$.

2.1 Matrix Branching Program

A branching program consists of several matrix chains and input functions with indices of input bit. To evaluate a matrix branching program, we multiply all matrices and output 0 or 1 depending on whether the product of the matrices is the same as a given matrix or not. We briefly review matrix branching programs.

Definition 1 (w-ary Matrix Branching Programs). *Let $\boldsymbol{A}_0$ be a $d_1 \times d_{\ell+1}$ matrix and w, ℓ, d, and N be natural numbers. A w-ary matrix branching program BP with length ℓ over N-bit inputs consists of the following data; a set of input functions $\{\mathsf{inp}_\mu : [\ell] \to [N]\}_{\mu \in [w]}$, a set of matrices $\{\boldsymbol{M}_{i,\boldsymbol{b}} \in$*

$\mathbb{Z}^{d_i \times d_{i+1}}\}_{i\in[\ell],\boldsymbol{b}\in\{0,1\}^w}$. *It has a domain for evaluations* $\{0,1\}^N$*, and evaluation of* BP *at* $\boldsymbol{x} = (x^v)_{v\in[w]}$ *is computed by*

$$BP(\boldsymbol{x}) = BP_{(\mathsf{inp}_\mu)_{\mu\in[w]},\boldsymbol{M}}(\boldsymbol{x}) = \begin{cases} 0 & \textit{if } \prod_{i=1}^{\ell} \boldsymbol{M}_{i,(x^\mu_{\mathsf{inp}_\mu(i)})_{\mu\in[w]}} = \boldsymbol{A}_0 \\ 1 & \textit{if } \prod_{i=1}^{\ell} \boldsymbol{M}_{i,(x^\mu_{\mathsf{inp}_\mu(i)})_{\mu\in[w]}} \neq \boldsymbol{A}_0 \end{cases}.$$

When w is set to 1 and ≥ 2, the matrix branching program is called a single-input and a multi-input matrix branching program, respectively. Throughout this paper, a matrix $\boldsymbol{A}_0$ is used as the zero matrix $\mathbf{0}$ or the identity matrix $\boldsymbol{I}_d$ if $d_i = d$ for all i. Moreover, we simplify the notation $(x^\mu_{\mathsf{inp}_\mu(i)})_{\mu\in[w]}$ as $\boldsymbol{x}_{\mathsf{inp}(i)}$.

Barrington proved all boolean functions can be expressed in the form of matrix branching program with bounded width [10]. The first candidate for *iO* [23] and following obfuscations [7, 15, 30, 32] exploit Barrington's theorem to transform circuits into BPs.

We also note that there are other methods to convert circuits into branching programs. Ben-Or and Cleve proved that the similar result to Barrington's theorem for arithmetic circuits [11]. Follow-up studies such as [3, 6] suggest more efficient methods for transformation. Their methods bypass the Barrington's theorem and make a circuit into a branching program directly. However, they still preserve the length of program, in other words, the length of branching program is equal to or larger than the size of circuit (number of gates).

We assume a mild condition on the branching programs: The length of branching program is $\Omega(N)$ for the number of input bits N. This is plausible since all input bits may affect the program, and the existing methods give much longer lengths. On the other hand, we do not restrict that the width/properties of the matrices in branching programs and the input function (such as single or dual input).

2.2 Indistinguishability Obfuscation

Definition 2 (Indistinguishability Obfuscation (*iO*)). *A PPT algorithm iO is an indistinguishability obfuscation for a circuit class* $\mathcal{C}$ *if the following conditions are satisfied:*

- *For all security parameters* $\lambda \in \mathbb{N}$*, for all circuits* $C \in \mathcal{C}$*, for all inputs* $\boldsymbol{x}$*, the following probability holds:*

$$\Pr\left[C'(\boldsymbol{x}) = C(\boldsymbol{x}) : C' \leftarrow iO(\lambda, C)\right] = 1.$$

- *For any PPT distinguisher* $\mathcal{D}$*, there exists a negligible function* α *satisfying the following statement: For all security parameters* $\lambda \in \mathbb{N}$ *and all pairs of circuits* C_0*,* $C_1 \in \mathcal{C}$*,* $C_0(\boldsymbol{x}) = C_1(\boldsymbol{x})$ *for all inputs* $\boldsymbol{x}$ *implies*

$$|\Pr\left[D(iO(\lambda, C_0)) = 1\right] - \Pr\left[D(iO(\lambda, C_1)) = 1\right]| \leq \alpha(\lambda).$$

Hereafter, we denote $iO(P)$ by an obfuscated program or obfuscation of a program, or a branching program P.

2.3 GGH13 Multilinear Map

Garg *et al.* suggest a candidate of multilinear map based on ideal lattice [22]. It is used to realize the indistinguishable obfuscation [23]. In this section, we briefly describe the GGH13 multilinear map. For more details, we recommend readers to refer [22]. Any parameters of multilinear maps are induced by the multilinearity parameter κ and the security parameters λ. For the sake of simplicity, we denote the multilinear maps which has the previous mentioned parameter as (κ, λ)-GGH multilinear map.

The multilinear map is sometimes called the graded encoding scheme. *i.e.*, All encodings of message have corresponding levels. Let $\boldsymbol{g}$ be a secret element in $\mathcal{R} = \mathbb{Z}[X]/\langle X^n+1\rangle$ and q a large integer. Then, the message space and encoding space are set by $\mathcal{M} = \mathcal{R}/\langle \boldsymbol{g}\rangle$ and $\mathcal{R}_q = \mathcal{R}/\langle q\rangle$, respectively. In order to represent a level of encodings, the set of secret invertible elements $\mathbb{L} = \{\boldsymbol{z}_i\}_{1\le i\le\kappa} \subset \mathcal{R}_q$ is chosen. We call a subset of $\mathbb{L}$ *level set* and elements in $\mathbb{L}$ *level parameters.*

For a small message $\boldsymbol{m} \in \mathcal{M}$, level-$L(\subset \mathbb{L})$ encoding of $\boldsymbol{m}$ is:

$$\mathsf{enc}_L(\boldsymbol{m}) = \left[\frac{\boldsymbol{r}\cdot\boldsymbol{g}+\boldsymbol{m}}{\prod_{i\in L}\boldsymbol{z}_i}\right]_q,$$

where $\boldsymbol{r} \in \mathcal{R}$ is a small random element. We call $\mathsf{enc}_{\mathbb{L}}(\boldsymbol{m})$, $\mathsf{enc}_{\{\boldsymbol{z}_i\}}(\boldsymbol{m})$ a top-level and level 1 encoding of $\boldsymbol{m}$, respectively. In addition, for a matrix $\boldsymbol{M}$, we denote a matrix whose entries are level-L encodings of corresponding entries of $\boldsymbol{M}$ by $\mathsf{enc}_L(\boldsymbol{M})$.

The arithmetic operations between encodings are defined as follows:

$$\begin{aligned}\mathsf{enc}_L(\boldsymbol{m}_1) + \mathsf{enc}_L(\boldsymbol{m}_2) &= \mathsf{enc}_L(\boldsymbol{m}_1+\boldsymbol{m}_2),\\ \mathsf{enc}_{L_1}(\boldsymbol{m}_1)\cdot\mathsf{enc}_{L_2}(\boldsymbol{m}_2) &= \mathsf{enc}_{L_1\sqcup L_2}(\boldsymbol{m}_1\cdot\boldsymbol{m}_2).\end{aligned}$$

Additionally, the (κ, λ)-GGH scheme provides a zerotesting parameter which can be used to determine whether a hidden message of a top-level encoding is zero or not. The zerotesting parameter $\boldsymbol{p}_{zt}$ is of the form:

$$\boldsymbol{p}_{zt} = \left[\boldsymbol{h}\cdot\frac{\prod_{i\in\mathbb{L}}\boldsymbol{z}_i}{\boldsymbol{g}}\right]_q,$$

where $\boldsymbol{h}$ is an $O(\sqrt{q})$-size element of $\mathcal{R}$. Given a top-level encoding of zero $\mathsf{enc}_{\mathbb{L}}(\boldsymbol{0}) = [\boldsymbol{r}\cdot\boldsymbol{g}/\prod_{i\in\mathbb{L}}\boldsymbol{z}_i]_q$, a zerotesting value is:

$$[\boldsymbol{p}_{zt}\cdot\mathsf{enc}_{\mathbb{L}}(\boldsymbol{0})]_q = \left[\boldsymbol{h}\cdot\frac{\prod_{i\in\mathbb{L}}\boldsymbol{z}_i}{\boldsymbol{g}}\cdot\frac{\boldsymbol{r}\cdot\boldsymbol{g}}{\prod_{i\in\mathbb{L}}\boldsymbol{z}_i}\right]_q = [\boldsymbol{h}\cdot\boldsymbol{r}]_q = \boldsymbol{h}\cdot\boldsymbol{r}\in\mathcal{R}.$$

We remark that a zerotesting value for a top-level encoding of nonzero gives an element of the form $[\boldsymbol{h}\cdot(\boldsymbol{r}+\boldsymbol{m}\cdot\boldsymbol{g}^{-1})]_q$, which is not small by Lemma 4 in [22]. Thus one can decide whether a message is zero or not by the zerotesting value.

Several papers [2,22,27] proposed the parameters of (κ, λ)-GGH13 multilinear map. Here we introduce the minimum conditions that satisfy the three works.

- $\log q = \tilde{\Theta}(\kappa \cdot \log n)$
- $n = \tilde{\Theta}(\kappa^{\epsilon} \cdot \lambda^{\delta})$ for constants δ, ϵ
- $M = \tilde{O}(n^{\Theta(1)})$

Here M is the size bound of numerators $\boldsymbol{r} \cdot \boldsymbol{g} + \boldsymbol{m}$ of level 1 encodings.[4] We note that the suggested parameters in [2, 27] choose $\delta = \epsilon = 1$, which enables the subexponential attack with respect to λ for small κ [1, 13]. When $\delta \geq 2$, all known direct attacks on GGH13 multilinear map require exponential time for classical adversary.

3 Main Theorem

In this section, we present the results from our attacks. We denote the obfuscation within our attack range as *the attackable obfuscation*, which is formally defined by *the attackable model* in the next section. The attackable obfuscation model encompasses all suggested BP obfuscations based on GGH13 multilinear map.

Proposition 1 (Universality of the Attackable Model) *BP obfuscations [3, 6, 7, 23, 24, 30, 32] satisfy all the constraints of the attackable model.*[5]

As a result, we obtain the following main theorem.

Theorem 1. *Let $\mathcal{O}$ be an attackable obfuscator, κ, λ be the multilinearity level and the security parameter of underlying GGH13 multilinear map. Suppose that the modulus q, dimension n, size bound M of numerators of level 1 encoding of underlying GGH13 satisfy $\log q = \tilde{\Theta}(\kappa \cdot \log n), M = \tilde{O}(n^{\Theta(1)})$. Then the following propositions hold:*

1. *For $n = \tilde{\Theta}(\kappa \cdot \lambda^{\delta})$ for a constant δ as in [2, 22, 27], there exist two functionally equivalent branching programs with $\Omega(\lambda^{\delta})$-length such that their obfuscated programs by $\mathcal{O}$ can be distinguished with high probability in polynomial time with respect to λ.*
2. *Moreover, for new parameter constraints $n = \tilde{\Theta}(\kappa^{\epsilon} \cdot \lambda^{\delta})$ for constants $\epsilon < 2, \delta$, there exist two functionally equivalent branching programs with $\Omega(\lambda^{\delta/(2-e)})$-length such that their obfuscated programs by $\mathcal{O}$ can be distinguished with high probability in polynomial time with respect to λ.*

The main theorem is proven by combining *converting program technique* and *matrix zeroizing attack* which are described in Section 5, 6. The bottleneck of the attack is the algorithm for NTRU, which is exploited in the middle step of converting technique; the other process can be done in polynomial time, while the time complexity to solve the NTRU problem relies on the parameters. The detailed analysis for the time complexity will be discussed in Section 5.3.

[4] The coefficients of random values are usually sampled from the Gaussian distribution. This do not hurt the result of this paper because the coefficients are bounded with overwhelming probability.

[5] We deal with easier model in the main body for simplicity. We can extend the model to capture the construction in [15]. This extended model is placed in Appendix A.

4 Attackable BP Obfuscations

In this section, we present a new BP obfuscation model which is attackable by our attack, *the attackable model.* We call a BP obfuscation captured by our model an *attackable BP obfuscation.*

The attackable model is composed of two steps; for a given BP, randomize BP, and encode randomized BPs by GGH13 multilinear map. More precisely, for a given branching program BP of the form

$$P = \left\{ \boldsymbol{M}_{i,\boldsymbol{b}} \in \mathbb{Z}^{d_i \times d_{i+1}} \right\}_{i \in [\ell], \boldsymbol{b} \in \{0,1\}^w},$$

we randomize P by several methods satisfying Definition 3 which will be described later. And then we encode each entries of randomized matrices and outputs the obfuscated program as the set

$$\begin{aligned} \mathcal{O}(P) = & \left\{ \widetilde{\boldsymbol{S}}, \widetilde{\boldsymbol{S}}' \in \mathcal{R}_q^{d_0 \times (d_1 + e_1)} \right\} \\ & \cup \left\{ \{ \widetilde{\boldsymbol{M}}_{i,\boldsymbol{b}}, \widetilde{\boldsymbol{M}}'_{i,\boldsymbol{b}} \in \mathcal{R}_q^{(d_i + e_i) \times (d_{i+1} + e_{i+1})} \}_{i \in [\ell], \boldsymbol{b} \in \{0,1\}^w}, \right\} \\ & \cup \left\{ \widetilde{\boldsymbol{T}}, \widetilde{\boldsymbol{T}}' \in \mathcal{R}_q^{(d_{\ell+1} + e_{\ell+1}) \times d_{\ell+2}} \right\} \end{aligned}$$

and the public parameters of GGH13 multilinear map. $\boldsymbol{S}, \boldsymbol{T}$ denote bookend matrices, and matrices with apostrophe mean the matrices of dummy program. In the attackable model, we specify the following property instead of establishing how to evaluate the program exactly. To evaluate the input value, a new function $Eval_{\widetilde{\boldsymbol{M}}} : \{0,1\}^N \to \mathcal{R}_q^{d_0 \times d_{\ell+2}}$ is computed as follows:

$$Eval_{\widetilde{\boldsymbol{M}}}(\boldsymbol{x}) = \widetilde{\boldsymbol{S}} \cdot \prod_{i=1}^{\ell} \widetilde{\boldsymbol{M}}_{i, \boldsymbol{x}_{\mathsf{inp}(i)}} \cdot \widetilde{\boldsymbol{T}} - \widetilde{\boldsymbol{S}}' \cdot \prod_{i=1}^{\ell} \widetilde{\boldsymbol{M}}'_{i, \boldsymbol{x}_{\mathsf{inp}(i)}} \cdot \widetilde{\boldsymbol{T}}' \in \mathcal{R}_q^{d_0 \times d_{\ell+2}}.$$

Proposition 2 (Evaluation of Obfuscation) *For a program P and program $\mathcal{O}(P)$ obfuscated by the attackable model, the evaluation of $\mathcal{O}(P)$ at a root $\boldsymbol{x}$ of P yields a top-level GGH13 encoding of zero in specific entry of the matrix $Eval_{\widetilde{\boldsymbol{M}}}(\boldsymbol{x})$. In other words, there are two integers u, v such that $Eval_{\widetilde{\boldsymbol{M}}}(\boldsymbol{x})[u, v]$ is an encoding of zero at level $\mathbb{L}$ for every input $\boldsymbol{x}$ satisfying $P(\boldsymbol{x}) = 0$.*

In the rest of this section, we explain specified descriptions of the attackable model in Section 4.1 and 4.2, and present a constraint of BPs to execute our attack in Section 4.3.

4.1 Randomization for Attackable Obfuscation Model

We introduce the conditions for BP randomization of attackable obfuscation model. These conditions for randomization covers all of the BP randomization methods suggested in the first candidate *iO* [23] and its subsequent works [3,6,7, 24,30,32]. In other words, higher dimension embedding, scalar bundling, Kilian randomization, bookend matrices (vectors), and dummy programs are captured by the attackable conditions.

Definition 3 (Attackable Conditions for Randomization). *For a branching program* $P = \left\{\boldsymbol{M}_{i,\boldsymbol{b}} \in \mathbb{Z}^{d_i \times d_{i+1}}\right\}_{i\in[\ell],\boldsymbol{b}\in\{0,1\}^w}$*, the attackable randomized branching program is the set*

$$\begin{aligned} Rand(P) = &\left\{\boldsymbol{R_S}, \boldsymbol{R'_S} \in \mathbb{Z}^{d_0\times(d_1+e_1)}\right\} \\ &\cup \left\{\{\boldsymbol{R}_{i,\boldsymbol{b}}, \boldsymbol{R}'_{i,\boldsymbol{b}} \in \mathbb{Z}^{(d_i+e_i)\times(d_{i+1}+e_{i+1})}\}_{i\in[\ell],\boldsymbol{b}\in\{0,1\}^w},\right\} \\ &\cup \left\{\boldsymbol{R_T}, \boldsymbol{R'_T} \in \mathbb{Z}^{(d_{\ell+1}+e_{\ell+1})\times d_{\ell+2}}\right\} \end{aligned}$$

satisfying the following properties, where $d_0, d_{\ell+2}, e_i$*'s are integers.*
1. There exist matrices $\boldsymbol{S}_0, \boldsymbol{S}'_0 \in \mathbb{Z}^{d_0\times d_1}, \boldsymbol{T}_0, \boldsymbol{T}'_0 \in \mathbb{Z}^{d_\ell\times d_{\ell+1}}$ *and scalars* $\boldsymbol{\alpha_S}, \boldsymbol{\alpha'_S}$*,* $\boldsymbol{\alpha_T}, \boldsymbol{\alpha'_T}$*,* $\{\boldsymbol{\alpha}_{i,\boldsymbol{b}}, \boldsymbol{\alpha}'_{i,\boldsymbol{b}}\}_{i\in[\ell],\boldsymbol{b}\in\{0,1\}^w}$ *such that the following equations hold for all* $\{\boldsymbol{b}_i \in \{0,1\}^w\}_{i\in[\ell]}$*:*

$$\begin{aligned} \boldsymbol{R}_S \cdot \prod_{i=1}^{\ell} \boldsymbol{R}_{i,\boldsymbol{b}_i} \cdot \boldsymbol{R}_T &= \boldsymbol{\alpha_S} \cdot \prod_{i=1}^{\ell} \boldsymbol{\alpha}_{i,\boldsymbol{b}_i} \cdot \boldsymbol{\alpha_T} \cdot \left(\boldsymbol{S}_0 \cdot \prod_{i=1}^{\ell} \boldsymbol{M}_{i,\boldsymbol{b}_i} \cdot \boldsymbol{T}_0\right), \\ \boldsymbol{R}'_S \cdot \prod_{i=1}^{\ell} \boldsymbol{R}'_{i,\boldsymbol{b}_i} \cdot \boldsymbol{R}'_T &= \boldsymbol{\alpha'_S} \cdot \prod_{i=1}^{\ell} \boldsymbol{\alpha}'_{i,\boldsymbol{b}_i} \cdot \boldsymbol{\alpha'_T} \cdot \left(\boldsymbol{S}'_0 \cdot \prod_{i=1}^{\ell} \boldsymbol{M}'_{i,\boldsymbol{b}_i} \cdot \boldsymbol{T}'_0\right). \end{aligned}$$

2. The evaluation of randomized program is done by checking whether the fixed entries of $RP(\boldsymbol{x}) := \boldsymbol{R_S} \cdot \prod_{i=1}^{\ell} \boldsymbol{R}_{i,\boldsymbol{x}_{\mathsf{inp}(i)}} \cdot \boldsymbol{R_T} - \boldsymbol{R'_S} \cdot \prod_{i=1}^{\ell} \boldsymbol{R}'_{i,\boldsymbol{x}_{\mathsf{inp}(i)}} \cdot \boldsymbol{R'_T}$ *are zero or not. Especially, there are two integers* u, v *such that* $P(\boldsymbol{x}) = 0 \Rightarrow RP(\boldsymbol{x})[u,v] = 0$*.*

Matrices with apostrophe are called *dummy matrices*, $\boldsymbol{R_S}, \boldsymbol{R'_S}, \boldsymbol{R_T}, \boldsymbol{R'_T}$ bookend matrices (vectors), and α's *bundling scalars*. When some elements of $Rand(P)$ (or bundling scalars) are trivial elements, we say that there is no such element.

4.2 Encoding by Multilinear Map

After the randomization, we encode the randomized matrix branching program by GGH13 multilinear map. We stress that we *do not encode* dummy/bookend matrices if there are no dummy/bookends, respectively.

For each randomized matrices, $\boldsymbol{R}_{i,\boldsymbol{b}}, \boldsymbol{R}'_{i,\boldsymbol{b}}$ and randomized bookend matrices $\boldsymbol{R_S}, \boldsymbol{R'_S}, \boldsymbol{R_T}, \boldsymbol{R'_T}$, we obtain the encoded matrices $\mathsf{enc}_{L_{i,\boldsymbol{b}}}(\boldsymbol{R}_{i,\boldsymbol{b}})$ whose entries are encoding of corresponding entries of randomized matrix $\boldsymbol{R}_{i,\boldsymbol{b}}$. For brevity we write $\widetilde{\boldsymbol{M}}_{i,\boldsymbol{b}}$ to denote $\mathsf{enc}_{L_{i,\boldsymbol{b}}}(\boldsymbol{R}_{i,\boldsymbol{b}})$, and the other matrices $\widetilde{\boldsymbol{M}}'_{i,\boldsymbol{b}}$, $\widetilde{\boldsymbol{S}}$, $\widetilde{\boldsymbol{S}}'$, $\widetilde{\boldsymbol{T}}, \widetilde{\boldsymbol{T}}'$ are defined in similar manner.

Two conditions should hold in the attackable model

1. the evaluation of valid input is top-level, in other words, for all input $\boldsymbol{x}$, $\left(\cup_{i=1}^{\ell} L_{i,\boldsymbol{x}_{\mathsf{inp}(i)}}\right) \cup L_{\boldsymbol{S}} \cup L_{\boldsymbol{T}} = \mathbb{L}$ where $\mathbb{L}$ denotes top-level set,
2. the sizes of set L's are all similar, that is, there is a constant C such that $|L_{i,\boldsymbol{b}}|/|L_{j,\boldsymbol{b}'}| \leq C$ for all $i, j, \boldsymbol{b}, \boldsymbol{b}'$ and similar inequalities hold for $L_{\boldsymbol{S}}, L_{\boldsymbol{T}}$.

In practice, the level L's is determined by *the straddling set system* introduced in [7, 30], and these constructions satisfy our conditions. Using the condition 1 and Definition 3, Proposition 2 can be easily verified. We also note that the condition 2 implies $\ell = \Theta(\kappa)$, where κ is the level of underlying multilinear map.

4.3 Linear Relationally Inequivalent Branching Programs

At last, we explain the condition, *linear relationally inequivalence*, for branching programs of attackable BP obfuscation. This condition is used at the last section, but we note that there are several linear relationally inequivalence BPs as stated in Proposition 3.

To define the linear relationally inequivalence, we consider evaluations of invalid inputs of branching program and denote $\prod_{i=1}^{\ell} \boldsymbol{M}_{i,\boldsymbol{b}_i}$ by $\boldsymbol{M}(\boldsymbol{b})$ for $\boldsymbol{b} = (\boldsymbol{b}_1, \cdots, \boldsymbol{b}_\ell)$. We define linear relations of two BPs and the *linear relationally inequivalence* of BPs as

Definition 4 (Linear Relations of Branching Program). *For a given branching program*

$$P_{\boldsymbol{M}} = \left\{ \boldsymbol{M}_{i,\boldsymbol{b}} \in \mathbb{Z}^{d_i \times d_{i+1}} \right\}_{i \in [\ell], \boldsymbol{b} \in \{0,1\}^w},$$

the set of linear relations of $P_{\boldsymbol{M}}$ is

$$L_{\boldsymbol{M}} := \left\{ (q_{\boldsymbol{b}})_{\boldsymbol{b} \in \{0,1\}^{w \times \ell}} : \sum_{\boldsymbol{b} \in \{0,1\}^{w \times \ell}} q_{\boldsymbol{b}} \cdot \boldsymbol{M}(\boldsymbol{b}) = \boldsymbol{0}^{d_1 \times d_{\ell+1}} \right\}$$

Definition 5 (Linear Relationally Inequivalence). *We say that two branching programs $P_{\boldsymbol{M}}$ and $P_{\boldsymbol{N}}$ with the same length are linear relationally inequivalent if $L_{\boldsymbol{M}} \neq L_{\boldsymbol{N}}$.*

The set of linear relations of a given BP is easily computed by computing the kernel, considering BP matrices as vectors. It is clear that $L_{\boldsymbol{M}}$ is a lattice. We note that the set of linear relations of BP is not determined by the functionality of BP, and indeed it seems that they are irrelevant.

Further, one can observe that if $P_{\boldsymbol{M}}, P_{\boldsymbol{N}}$ are linear relationally inequivalent BPs, then so do two extended BPs $P'_{\boldsymbol{M}}, P'_{\boldsymbol{N}}$ which are obtained by concatenating some other (functionally equivalent) BPs on the right (or left) of $P_{\boldsymbol{M}}, P_{\boldsymbol{N}}$. Therefore we can show that there exist arbitrary large two functionally equivalent BPs which are linear relationally inequivalent.

We conclude this section by presenting a proposition that shows concrete examples of linear relationally inequivalent BPs, which are placed in Appendix C.

Proposition 3 *There are two functionally equivalent, but linear relationally inequivalent branching programs. Especially, there are examples satisfying the linear relationally inequivalence which are*
1) generated by Barrington's theorem and input-unpartitionable or
2) from non-deterministic finite automata and read-once, in other words, inp *is a bijection.*

5 Program Converting Technique

In this section, we describe the program converting technique, which remove the hindrance of modulus q and $\boldsymbol{g}$. We first define new notion $\boldsymbol{Y}$ *program (of P)* if all entries of branching program matrices corresponding a program P are in a space $\boldsymbol{Y}$ while preserving many properties. For example, the obfuscated program $\mathcal{O}(P)$ is $\mathcal{R}_q$ program. Suppose that the obfuscated program $\mathcal{O}(P)$ of program P is given.

We will convert given obfuscated program $\mathcal{O}(P)$ into $\mathcal{R}$ and $\mathcal{R}/\langle \boldsymbol{g} \rangle$ program using the algorithm to solve the NTRU problem, especially *subfield attacks* [1,18] which solves the problem with large modulus q.

Proposition 4 ([1, 17, 18, 26]) *Let q be a large integer, n a power of two, M a constant much smaller than q, $\mathcal{R} = \mathbb{Z}[X]/\langle X^n + 1 \rangle$ and $\mathcal{R}_q = \mathcal{R}/q\mathcal{R}$. For a given $[\boldsymbol{f}_1/\boldsymbol{f}_2]_q \in \mathcal{R}_q$ for $\boldsymbol{f}_1, \boldsymbol{f}_2 \in \mathcal{R}$ with size smaller than M, there is an algorithm to compute $(\boldsymbol{c} \cdot \boldsymbol{f}_2, \boldsymbol{c} \cdot \boldsymbol{f}_1) \in \mathcal{R}^2$ such that sizes of $\boldsymbol{c}$, $\boldsymbol{c} \cdot \boldsymbol{f}_1$ and $\boldsymbol{c} \cdot \boldsymbol{f}_2$ are much smaller than q in time $2^{O(\beta)} \cdot poly(n)$ for a constant β satisfying $\beta/\log\beta = \Theta(n \log M/\log^2 q)$.*

We note that the similar results hold for other non-cyclotomic ring [17,26] or for $\boldsymbol{f}_1, \boldsymbol{f}_2$ from certain distribution [1]. Throughout in this paper, we only consider the bounded coefficient $\boldsymbol{f}_1 \boldsymbol{f}_2$ in cyclotomic ring for brevity.

For given obfuscated program in $\mathcal{R}_q$, we first make the NTRU instances and solve the problem, and then convert to $\mathcal{R}$ program by some computations on obfuscated matrices. This procedure replaces the level parameter $\boldsymbol{z}_i$ with a small element $\boldsymbol{c}_i$. The $\mathcal{R}$ program preserves same functionality with the $\mathcal{R}_q$ program. Subsequently, we convert this $\mathcal{R}$ program to $\mathcal{R}/\langle \boldsymbol{g} \rangle$ program by recovering the ideal $\langle \boldsymbol{g} \rangle$.

5.1 Converting to $\mathcal{R}$ Program

In order to remove the modulus q, we employ the algorithm for solving NTRU problem. Let $\widetilde{\boldsymbol{M}}_{i,\boldsymbol{b}}$ be the obfuscated matrix of $\boldsymbol{R}_{i,\boldsymbol{b}}$. Then, each (j,k)-th entries of obfuscated matrix $\widetilde{\boldsymbol{M}}_{i,\boldsymbol{b}}$ is of the form

$$\boldsymbol{d}_{j,k,\boldsymbol{b}} = \left[\frac{\boldsymbol{r}_{j,k,\boldsymbol{b}} \cdot \boldsymbol{g} + \boldsymbol{a}_{j,k,\boldsymbol{b}}}{\boldsymbol{z}_i} \right]_q ,$$

where $\boldsymbol{a}_{j,k,\boldsymbol{b}}$ is the (j,k)-th entry of the matrix $\boldsymbol{R}_{i,\boldsymbol{b}}$ and $\boldsymbol{r}_{j,k,\boldsymbol{b}} \in \mathcal{R}$ are random small elements. Consider an element $\boldsymbol{v} = [\boldsymbol{d}_{1,1,\mathbf{0}}/\boldsymbol{d}_{1,2,\mathbf{0}}]_q = [(\boldsymbol{r}_{1,1,\mathbf{0}} \cdot \boldsymbol{g} + \boldsymbol{a}_{1,1,\mathbf{0}})/(\boldsymbol{r}_{1,2,\mathbf{0}} \cdot \boldsymbol{g} + \boldsymbol{a}_{1,2,\mathbf{0}})]_q$. Then, $\boldsymbol{v}$ is the instance of the NTRU problem since the size of denominator and numerator of $\boldsymbol{v}$ is much smaller than q in the parameter setup of GGH13 multilinear map.

Applying Proposition 4 to an instance $\boldsymbol{v}$, one can find a pair $(\boldsymbol{c}_i \cdot (\boldsymbol{r}_{1,1,\mathbf{0}} \cdot \boldsymbol{g} + \boldsymbol{a}_{1,1,\mathbf{0}}),\ \boldsymbol{c}_i \cdot (\boldsymbol{r}_{1,2,\mathbf{0}} \cdot \boldsymbol{g} + \boldsymbol{a}_{1,2,\mathbf{0}})) \in \mathcal{R}^2$ with relatively small $\boldsymbol{c}_i \in \mathcal{R}$. Further, for any element $\boldsymbol{d}_{j,k,\boldsymbol{b}} \in \widetilde{\boldsymbol{M}}_{i,\boldsymbol{b}}$, we can remove the modulus q by computing

$$\boldsymbol{c}_i \cdot (\boldsymbol{r}_{1,1,\mathbf{0}} \cdot \boldsymbol{g} + \boldsymbol{a}_{1,1}, \mathbf{0}) \cdot [\boldsymbol{d}_{j,k,\boldsymbol{b}}/\boldsymbol{d}_{1,1,\mathbf{0}}]_q = \boldsymbol{c}_i \cdot (\boldsymbol{r}_{j,k,\mathbf{0}} \cdot \boldsymbol{g} + \boldsymbol{a}_{j,k,\mathbf{0}}) \in \mathcal{R}$$

because of the small size of $\boldsymbol{c}_i$. Consequently, one can obtain a new matrix $\boldsymbol{D}_{i,\boldsymbol{b}}$ over $\mathcal{R}$ whose (j,k)-th entry is $\boldsymbol{c}_i \cdot (\boldsymbol{r}_{j,k,\boldsymbol{0}} \cdot \boldsymbol{g} + \boldsymbol{a}_{j,k,\boldsymbol{0}})$.

Similarly, a new dummy matrix $\boldsymbol{D}'_{i,\boldsymbol{b}}$ over $\mathcal{R}$ can be obtained because $\widetilde{\boldsymbol{M}}'_{i,\boldsymbol{b}}$ shares the level parameter $\boldsymbol{z}_i$ with $\widetilde{\boldsymbol{M}}_{i,\boldsymbol{b}}$ by multiplying $\boldsymbol{c}_i \cdot (\boldsymbol{r}_{j,k,\boldsymbol{0}} \cdot \boldsymbol{g} + \boldsymbol{a}_{j,k,\boldsymbol{0}})$ to $[\boldsymbol{d}'_{j,k,\boldsymbol{b}}/\boldsymbol{d}_{1,1,\boldsymbol{0}}]_q$ where $\boldsymbol{d}'_{j,k,\boldsymbol{b}}$ is a (j,k)-th entry of $\widetilde{\boldsymbol{S}}'_{i,\boldsymbol{b}}$. We easily observe that $2 \cdot 2^w$ matrices $\boldsymbol{D}_{i,\boldsymbol{b}}$ and $\boldsymbol{D}'_{i,\boldsymbol{b}}$ share the parameter $\boldsymbol{c}_i$.

For all matrices $\widetilde{\boldsymbol{M}}_{i,\boldsymbol{b}}$ and $\widetilde{\boldsymbol{M}}'_{i,\boldsymbol{b}}$ with $i \in [\ell]$ and $\boldsymbol{b} \in \{0,1\}^w$, we can obtain new matrices $\boldsymbol{D}_{i,\boldsymbol{b}}$ and $\boldsymbol{D}'_{i,\boldsymbol{b}}$ over $\mathcal{R}$. In the case of bookend matrices $\widetilde{\boldsymbol{S}}$ and $\widetilde{\boldsymbol{T}}$, they are converted into matrices over $\mathcal{R}$ with small constants $\boldsymbol{c}_{\boldsymbol{S}}$ and $\boldsymbol{c}_{\boldsymbol{T}}$, respectively. Note that this step runs in polynomial time if κ is large [1,17,18,26]. Detailed analysis of this part is discussed in Section 5.3.

Therefore, we can convert $\mathcal{R}_q$-program $\mathcal{O}(P)$ into a new program, *$\mathcal{R}$-program of P*:

$$\mathcal{R}(P) = \{\boldsymbol{D}_{\boldsymbol{S}}, \boldsymbol{D}_{\boldsymbol{T}}, \boldsymbol{D'}_{\boldsymbol{S}}, \boldsymbol{D'}_{\boldsymbol{T}}, \{\boldsymbol{D}_{i,\boldsymbol{b}}, \boldsymbol{D'}_{i,\boldsymbol{b}}\}_{i\in[\ell],\boldsymbol{b}\in\{0,1\}^w}\}.$$

Note that the matrix $\boldsymbol{D}_{i,\boldsymbol{b}}$ of $\mathcal{R}(P)$ is of the form $\boldsymbol{c}_i \cdot \boldsymbol{R}_{i,\boldsymbol{b}} \pmod{\langle \boldsymbol{g} \rangle}$ in $\mathcal{R}/\langle \boldsymbol{g} \rangle$.

Dummy and bookend matrices satisfies similar relations. We denote $\boldsymbol{c}_i \cdot \boldsymbol{\alpha}_{i,\boldsymbol{b}}$ and $\boldsymbol{c}_i \cdot \boldsymbol{\alpha}'_{i,\boldsymbol{b}}$ by $\boldsymbol{\rho}_{i,\boldsymbol{b}}$, $\boldsymbol{\rho}'_{i,\boldsymbol{b}}$ for simplicity. The properties of Definition 3 is naturally extended to the following. The proposition 5 means an evaluation of $\mathcal{R}(P)$ preserves the functionality up to constant on the valid input $\boldsymbol{x}$.

Proposition 5 (Evaluation of $\mathcal{R}$ and $\mathcal{R}/\langle \boldsymbol{g} \rangle$ Branching Program) *For a $\mathcal{R}$ program given in this section, the following propositions holds:*
1. The higher dimension embedding matrices $\boldsymbol{U}$'s are eliminated in the product of randomized matrix branching program, that is, there are matrices $\boldsymbol{S}_0, \boldsymbol{S}'_0 \in \mathbb{Z}^{d_0 \times d_1}, \boldsymbol{T}_0, \boldsymbol{T}'_0 \in \mathbb{Z}^{d_{\ell+1} \times d_{\ell+2}}$ such that the following equations hold for all input x:

$$\boldsymbol{D}_{\boldsymbol{S}} \cdot \prod_{i=1}^{\ell} \boldsymbol{D}_{i,\boldsymbol{b}_i} \cdot \boldsymbol{D}_{\boldsymbol{T}} = \boldsymbol{\rho}_{\boldsymbol{S}} \cdot \prod_{i=1}^{\ell} \boldsymbol{\rho}_{i,\boldsymbol{b}_i} \cdot \boldsymbol{\rho}_{\boldsymbol{T}} \cdot \left(\boldsymbol{S}_0 \cdot \prod_{i=1}^{\ell} \boldsymbol{M}_{i,\boldsymbol{b}_i} \cdot \boldsymbol{T}_0 \right) \pmod{\langle \boldsymbol{g} \rangle},$$
$$\boldsymbol{D}'_{\boldsymbol{S}} \cdot \prod_{i=1}^{\ell} \boldsymbol{D}'_{i,\boldsymbol{b}_i} \cdot \boldsymbol{D}'_{\boldsymbol{T}} = \boldsymbol{\rho}'_{\boldsymbol{S}} \cdot \prod_{i=1}^{\ell} \boldsymbol{\rho}'_{i,\boldsymbol{b}_i} \cdot \boldsymbol{\rho}'_{\boldsymbol{T}} \cdot \left(\boldsymbol{S}'_0 \cdot \prod_{i=1}^{\ell} M'_{i,\boldsymbol{b}_i} \cdot \boldsymbol{T}'_0 \right) \pmod{\langle \boldsymbol{g} \rangle}.$$

2. The evaluation of $\mathcal{R}$ program is done by checking whether the fixed entries of $Eval_{\boldsymbol{D}}(\boldsymbol{x}) := \boldsymbol{D}_{\boldsymbol{S}} \cdot \prod_{i=1}^{\ell} \boldsymbol{D}_{i,\boldsymbol{x}_{\mathsf{inp}(i)}} \cdot \boldsymbol{D}_{\boldsymbol{T}} - \boldsymbol{D'}_{\boldsymbol{S}} \cdot \prod_{i=1}^{\ell} \boldsymbol{D'}_{i,\boldsymbol{x}_{\mathsf{inp}(i)}} \cdot \boldsymbol{D'}_{\boldsymbol{T}}$ is multiple of $\boldsymbol{g}$ or not. Especially, there are two integers u, v such that $P(\boldsymbol{x}) = 0 \Rightarrow Eval_{\boldsymbol{D}}(\boldsymbol{x})[u,v] = 0 \pmod{\langle \boldsymbol{g} \rangle}$)

5.2 Recovering $\langle g \rangle$ and Converting to $\mathcal{R}/\langle g \rangle$ Program

Next, we will compute a basis of the plaintext space $\langle \boldsymbol{g} \rangle$ to transform $\mathcal{R}$ program into $\mathcal{R}/\langle \boldsymbol{g} \rangle$-program. Unlike other attacks, we do not use the assumption 'input partitionability'. We exploits the fact that $\mathcal{R}$ program which comes from $\mathcal{R}_q$

program has the same functionality up to constant. However, existing attacks with input partitionable assumption and our cryptanalysis cannot be applied to a BP program for an 'evasive function' since it does not output multiples of $\boldsymbol{g}$. It consists of following two steps:

Finding a multiple of $\boldsymbol{g}$. This step is done by computing $Eval_{\boldsymbol{D}}$ at the zeros of program P. We compute $Eval_{\boldsymbol{D}}(\boldsymbol{x})$ for $\mathcal{R}$ program $\mathcal{R}(P)$ at $\boldsymbol{x}$ satisfying $P(\boldsymbol{x}) = 0$. Then, Proposition 5 implies that $Eval_{\boldsymbol{D}}(\boldsymbol{x})[u, v]$ is a multiple of $\boldsymbol{g}$. More precisely, $Eval_{\boldsymbol{D}}(\boldsymbol{x})[u, v]$ is of the form

$$\boldsymbol{c_S} \cdot \boldsymbol{c_T} \cdot \prod_{i=1}^{\ell} \boldsymbol{c}_i \cdot \boldsymbol{a} \cdot \boldsymbol{g}$$

when $\boldsymbol{p}_{zt} \cdot Eval_{\widetilde{\boldsymbol{M}}}(\boldsymbol{x})[u, v] = \boldsymbol{a} \cdot \boldsymbol{h} \pmod q$ for some $\boldsymbol{a} \in \mathcal{R}$ such that $\|\boldsymbol{a} \cdot \boldsymbol{h}\|_2$ is less than $q^{3/4}$.

This procedure outputs the value which is not only multiple of $\boldsymbol{g}$ but also $\boldsymbol{c}_i$'s. However, we can generate several different $\mathcal{R}$ program from $\mathcal{O}(P)$ for different solutions of Proposition 4. We assume that the multiples of $\boldsymbol{g}$ from different $\mathcal{R}$ program are independent multiples of $\boldsymbol{g}$, with the randomized lattice reduction algorithm as in [21].

Computing Hermite Normal Form of $\langle \boldsymbol{g} \rangle$. For given several random multiples $\boldsymbol{f}_i \cdot \boldsymbol{g}$ of $\boldsymbol{g}$, we can recover a basis of $\langle \boldsymbol{g} \rangle$ by computing sum of sufficiently many ideal $\langle \boldsymbol{f} \cdot \boldsymbol{g} \rangle$ represented by a lattice with basis $\{\boldsymbol{f} \cdot \boldsymbol{g}, \boldsymbol{f} \cdot \boldsymbol{g} \cdot X, \cdots, \boldsymbol{f} \cdot \boldsymbol{g} \cdot X^{n-1}\}$ or computing the Hermite Normal Form of union of their generating sets by applying the lemma [1, Lemma 1].

Both computations are done in polynomial time in λ and κ, since the evaluations and computing the Hermite normal form has a polynomial time complexity. Eventually, we recover the basis of ideal lattice $\langle \boldsymbol{g} \rangle$ and we can efficiently compute the arithmetics in $\mathcal{R}/\langle \boldsymbol{g} \rangle$. In other words, we get a $\mathcal{R}/\langle \boldsymbol{g} \rangle$ program corresponding to $\mathcal{O}(P)$ (or P), whose properties are characterized by Proposition 5. For convenience, we abuse the notation; from now, $\mathcal{R}(P)$ is the $\mathcal{R}/\langle \boldsymbol{g} \rangle$ program and $\boldsymbol{D_S}, \boldsymbol{D_T}$ and $\boldsymbol{D}_{i,\boldsymbol{b}}$ for all $i \in [\ell], \boldsymbol{b} \in \{0,1\}^w$ are matrices over $\mathcal{R}/\langle \boldsymbol{g} \rangle$.

5.3 Analysis of the Converting Technique

We discuss the time complexity of our program converting technique. The program converting consists of converting to $\mathcal{R}$ program, evaluating of $\mathcal{R}$ program, computing a Hermite Normal Form of an ideal lattice $\langle \boldsymbol{g} \rangle$. The last two steps take polynomial time complexity, so the total cost is dominated by the first step. More precisely, solving the NTRU problem for each encoded matrix is the dominant part of the program converting.

To estimate the cost of solving the NTRU problem, we assume that each component of branching program is encoded by GGH13 multilinear map in level-1. The general cases are similar but a bit more complex when we assume that the size of level sets are not too different so that $\ell = \Theta(\kappa)$.

Suppose that an obfuscated branching program $\mathcal{O}(P)$ over (κ, λ)-GGH13 multilinear map is given. As we written in Section 2.3, for constants δ, e and security parameter λ, multilinearity level κ, n, M, and $\log q$ are set to be $\tilde{\Theta}(\kappa^e \cdot \lambda^\delta)$, $n^{\Theta(1)}$, and $\tilde{\Theta}(\kappa \cdot \log n)$, respectively. Proposition 4 implies that one can convert the program in $2^{O(\beta)} \cdot poly(\lambda, \kappa)$ time for $\frac{\beta}{\log \beta} = \Theta(\frac{n \log M}{\log^2 q}) = \tilde{\Theta}\left(\frac{\lambda^\delta}{\kappa^{2-e}}\right)$. Therefore, the program converting technique is done in polynomial time for $\kappa = \tilde{\Omega}(\lambda^{\delta/(2-e)})$. Alternatively, the program converting technique is done in polynomial time for obfuscated programs with length $\ell = \tilde{\Omega}(\lambda^{\delta/(2-e)})$.

We note that choosing large n to make the subfield attack work in exponential time rules out our attack as well. More concretly, if one chooses $n = \tilde{\Theta}(\kappa^2 \lambda)$ then the underlying NTRU problem is hard enough to block known subexponential time attacks.

6 Matrix Zeroizing Attack

In this section, we present a distinguishing attack on $\mathcal{R}$ programs to complete our cryptanalysis of attackable BP obfuscation model. We note that we can evaluate the $\mathcal{R}$ program at invalid inputs, or *mixed input*, since the multilinearity level which was the obstacle of mixed inputs is removed in the previous step. We recall that $\boldsymbol{M}(\boldsymbol{b})$ denotes $\prod_{i=1}^{\ell} \boldsymbol{M}_{i,\boldsymbol{b}_i}$ for $\boldsymbol{b} = (\boldsymbol{b}_1, \cdots, \boldsymbol{b}_\ell)$ and the set of linear relations

$$L_{\boldsymbol{M}} = \left\{ (q_{\boldsymbol{b}})_{\boldsymbol{b} \in \{0,1\}^{w \times \ell}} : \sum_{\boldsymbol{b} \in \{0,1\}^{w \times \ell}} q_{\boldsymbol{b}} \cdot \boldsymbol{M}(\boldsymbol{b}) = \boldsymbol{0}^{d_1 \times d_{\ell+1}} \right\}$$

which was defined in Section 4.3. We also recall that the two program $\boldsymbol{M}$ and $\boldsymbol{N}$ are linear relationally inequivalent if $L_{\boldsymbol{M}} \neq L_{\boldsymbol{N}}$.

For two functionally equivalent but linear relationally inequivalent BPs $P_{\boldsymbol{M}}$ and $P_{\boldsymbol{N}}$, we will zeroize the $\boldsymbol{R}$ program corresponding to $P_{\boldsymbol{M}}$ by exploiting the linear relation, whereas $\boldsymbol{R}$ program corresponding to $P_{\boldsymbol{N}}$ would not be a zero matrix. The result of the matrix zeroizing attack is as follows.

Proposition 6 (Matrix Zeroizing Attack) *For functionally equivalent but linear relationally inequivalent branching programs $P_{\boldsymbol{M}}, P_{\boldsymbol{N}}$, there is a PPT algorithm which can distinguish between two $\mathcal{R}$ programs $\mathcal{R}(P_{\boldsymbol{M}})$ and $\mathcal{R}(P_{\boldsymbol{N}})$ obtained by the method in Section 5 with non-negligible probability.*

Now we explain how to distinguish two $\mathcal{R}$ programs using linear relationally inequivalence. Despite the absence of multilinearity level, we still have obstacles to directly exploit linear relationally inequivalence: scalar bundlings. To explain the main idea of the attack, we assume that, for the time being, all scalar bundling are trivial in the obtained program in Section 5. We later explain how to deal the scalar bundlings.

Suppose that two BPs $P_{\boldsymbol{M}}, P_{\boldsymbol{N}}$ and an $\boldsymbol{R}$ program

$$\mathcal{R}(P_{\boldsymbol{X}}) = \{\boldsymbol{D}_{\boldsymbol{S}}, \boldsymbol{D}_{\boldsymbol{T}}, \boldsymbol{D}_{\boldsymbol{S}'}, \boldsymbol{D}_{\boldsymbol{T}'}, \{\boldsymbol{D}_{i,\boldsymbol{b}}, \boldsymbol{D'}_{i,\boldsymbol{b}}\}_{i \in [\ell], \boldsymbol{b} \in \{0,1\}^w}\}$$

are given. Our goal is to determine $\boldsymbol{X} = \boldsymbol{N}$ or $\boldsymbol{X} = \boldsymbol{M}$. We can compute a linear relation $(q_{\boldsymbol{b}})$ which is an element of $L_{\boldsymbol{M}} \setminus L_{\boldsymbol{N}}$ in polynomial time[6] by computing a basis of kernel, and solve the membership problems of lattice for each vector in the basis. Then the following equation holds

$$\sum_{\boldsymbol{b}\in\{0,1\}^{w\times\ell}} \left(q_{\boldsymbol{b}} \cdot \boldsymbol{D}_{\boldsymbol{S}} \cdot \prod_{i=1}^{\ell} \boldsymbol{D}_{i,\boldsymbol{b}_i} \cdot \boldsymbol{D}_{\boldsymbol{T}} \right) \quad = \sum_{\boldsymbol{b}\in\{0,1\}^{w\times\ell}} \left(q_{\boldsymbol{b}} \cdot \boldsymbol{S}_0 \cdot \prod_{i=1}^{\ell} \boldsymbol{M}_{i,\boldsymbol{b}_i} \cdot \boldsymbol{T}_0 \right)$$

$$= \boldsymbol{S}_0 \cdot \sum_{\boldsymbol{b}\in\{0,1\}^{w\times\ell}} \left(q_{\boldsymbol{b}} \cdot \prod_{i=1}^{\ell} \boldsymbol{M}_{i,\boldsymbol{b}_i} \right) \cdot \boldsymbol{T}_0 \quad = \boldsymbol{S}_0 \cdot \boldsymbol{0}^{d_1 \times d_{\ell+1}} \cdot \boldsymbol{T}_0 = \boldsymbol{0}^{d_0 \times d_{\ell+2}} \pmod{\langle \boldsymbol{g} \rangle}$$

when $\boldsymbol{X} = \boldsymbol{M}$ whereas this is not hold when $\boldsymbol{X} = \boldsymbol{N}$. Therefore, the matrix zeroizing attack works when the scalar bundlings are all trivial.

When the scalar bundlings are not trivial, we can do the similar computation after recovering ratios of bundling scalars. Assume that we know $\boldsymbol{\rho}_{i,\boldsymbol{u}}/\boldsymbol{\rho}_{i,\boldsymbol{v}}$ for every $1 \le i \le \ell$ and $\boldsymbol{u}, \boldsymbol{v} \in \{0,1\}^w$. Consequently, for $\boldsymbol{r}(\boldsymbol{b}) := \prod_{i\in[\ell]} \boldsymbol{\rho}_{i,\boldsymbol{b}_i}$ where $\boldsymbol{b} = (\boldsymbol{b}_1, \cdots, \boldsymbol{b}_\ell)$, we can compute $\boldsymbol{r}(\boldsymbol{b})/\boldsymbol{r}(\boldsymbol{c})$ for $\boldsymbol{b}, \boldsymbol{c} \in \{0,1\}^{w\times\ell}$ by multiplying ratios of bundling scalars. Then, we can calculate

$$\sum_{\boldsymbol{b}\in\{0,1\}^{w\times\ell}} \left(q_{\boldsymbol{b}} \cdot \frac{\boldsymbol{r}(\boldsymbol{0})}{\boldsymbol{r}(\boldsymbol{b})} \cdot \boldsymbol{D}_{\boldsymbol{S}} \cdot \prod_{i=1}^{\ell} \boldsymbol{D}_{i,\boldsymbol{b}_i} \cdot \boldsymbol{D}_{\boldsymbol{T}} \right)$$

$$= \sum_{\boldsymbol{b}\in\{0,1\}^{w\times\ell}} \left(q_{\boldsymbol{b}} \cdot \boldsymbol{\rho}_{\boldsymbol{S}} \cdot \boldsymbol{r}(\boldsymbol{0}) \cdot \boldsymbol{\rho}_{\boldsymbol{T}} \cdot \boldsymbol{S}_0 \cdot \prod_{i=1}^{\ell} \boldsymbol{M}_{i,\boldsymbol{b}_i} \cdot \boldsymbol{T}_0 \right)$$

$$= \boldsymbol{\rho}_{\boldsymbol{S}} \cdot \boldsymbol{r}(\boldsymbol{0}) \cdot \boldsymbol{\rho}_{\boldsymbol{T}} \cdot \boldsymbol{S}_0 \cdot \sum_{\boldsymbol{b}\in\{0,1\}^{w\times\ell}} \left(q_{\boldsymbol{b}} \cdot \prod_{i=1}^{\ell} \boldsymbol{M}_{i,\boldsymbol{b}_i} \right) \cdot \boldsymbol{T}_0 \pmod{\langle \boldsymbol{g} \rangle},$$

which is a zero matrix if and only if $\boldsymbol{X} = \boldsymbol{M}$.

Accordingly, we should remove the scalar bundlings or recover ratios of scalar bundlings to execute the matrix zeroizing attack. In the rest of this section, we show how to recover or remove (ratios of) scalar bundlings in several cases. In Section 6.2, we explain how to recover all ratios in general cases by complex techniques.

6.1 Existing BP Obfuscations

In this section, we show how to apply the matrix zeroizing attack on two remarkable obfuscations, GGHRSW and GMMSSZ. The other examples on obfuscations [6, 32] are placed in Appendix B.

[6] The dimension of $(q_{\boldsymbol{b}})_{\boldsymbol{b}\in\{0,1\}^{w\times\ell}}$ is $2^{w\times\ell}$, which is exponentially large. However, we can reduce this exponential part by considering a polynomial number of $\boldsymbol{b}$ so that there are linear relations.

GGHRSW. As the first case, we consider the first BP obfuscation, GGHRSW, which has the identity dummy program. We note that the attack for this case works for the attackable BP obfuscations with fixed dummy program as well. For this case, a constraint on the bundling scalars $\boldsymbol{\alpha}_{\boldsymbol{x}} = \boldsymbol{\alpha}'_{\boldsymbol{x}}$ for every input $\boldsymbol{x}$ is given where $\boldsymbol{\alpha}_{\boldsymbol{x}} = \boldsymbol{\alpha}_{\boldsymbol{S}} \cdot \prod_{i=1}^{\ell} \boldsymbol{\alpha}_{i,\boldsymbol{x}_{\mathsf{inp}(i)}} \cdot \boldsymbol{\alpha}_{\boldsymbol{T}},\ \boldsymbol{\alpha}'_{\boldsymbol{x}} = \boldsymbol{\alpha}'_{\boldsymbol{S}} \cdot \prod_{i=1}^{\ell} \boldsymbol{\alpha}'_{i,\boldsymbol{x}_{\mathsf{inp}(i)}} \cdot \boldsymbol{\alpha}'_{\boldsymbol{T}}$. Suppose $\mathcal{R}$ *program of* P is given by

$$\mathcal{R}(P) = \{\boldsymbol{D}_{\boldsymbol{S}}, \boldsymbol{D}_{\boldsymbol{T}}, \boldsymbol{D}_{\boldsymbol{S}'}, \boldsymbol{D}_{\boldsymbol{T}'}, \{\boldsymbol{D}_{i,\boldsymbol{b}}, \boldsymbol{D}'_{i,\boldsymbol{b}}\}_{i\in[\ell],\boldsymbol{b}\in\{0,1\}^w}\}.$$

By Proposition 5, the following equations hold

$$\boldsymbol{D}_{\boldsymbol{S}} \cdot \prod_{i=1}^{\ell} \boldsymbol{D}_{i,\boldsymbol{x}_{\mathsf{inp}(i)}} \cdot \boldsymbol{D}_{\boldsymbol{T}} = \boldsymbol{\rho}_{\boldsymbol{S}} \cdot \prod_{i=1}^{\ell} \boldsymbol{\rho}_{i,\boldsymbol{x}_{\mathsf{inp}(i)}} \cdot \boldsymbol{\rho}_{\boldsymbol{T}} \cdot \left(\boldsymbol{S}_0 \cdot \prod_{i=1}^{\ell} \boldsymbol{M}_{i,\boldsymbol{x}_{\mathsf{inp}(i)}} \cdot \boldsymbol{T}_0 \right) \bmod \langle \boldsymbol{g} \rangle,$$

$$\boldsymbol{D}'_{\boldsymbol{S}} \cdot \prod_{i=1}^{\ell} \boldsymbol{D}'_{i,\boldsymbol{x}_{\mathsf{inp}(i)}} \cdot \boldsymbol{D}'_{\boldsymbol{T}} = \boldsymbol{\rho}'_{\boldsymbol{S}} \cdot \prod_{i=1}^{\ell} \boldsymbol{\rho}'_{i,\boldsymbol{x}_{\mathsf{inp}(i)}} \cdot \boldsymbol{\rho}'_{\boldsymbol{T}} \cdot \left(\boldsymbol{S}'_0 \cdot \prod_{i=1}^{\ell} \boldsymbol{M}'_{i,\boldsymbol{x}_{\mathsf{inp}(i)}} \cdot \boldsymbol{T}'_0 \right) \bmod \langle \boldsymbol{g} \rangle.$$

Here we assume that each $\boldsymbol{M}'_{i,\boldsymbol{x}_{\mathsf{inp}(i)}}$ are identity matrices. Now we consider the two quantity of evaluations $Plain_{\boldsymbol{D}}(\boldsymbol{x}) := \boldsymbol{D}_{\boldsymbol{S}} \cdot \prod_{i=1}^{\ell} \boldsymbol{D}_{i,\boldsymbol{x}_{\mathsf{inp}(i)}} \cdot \boldsymbol{D}_{\boldsymbol{T}}$ and $Dummy_{\boldsymbol{D}}(\boldsymbol{x}) := \boldsymbol{D'}_{\boldsymbol{S}} \cdot \prod_{i=1}^{\ell} \boldsymbol{D'}_{i,\boldsymbol{x}_{\mathsf{inp}(i)}} \cdot \boldsymbol{D'}_{\boldsymbol{T}}$.

According to the condition of scalar bundlings, $\boldsymbol{\rho}_{\boldsymbol{S}} \cdot \prod_{i=1}^{\ell} \boldsymbol{\rho}_{i,\boldsymbol{x}_{\mathsf{inp}(i)}} \cdot \boldsymbol{\rho}_{\boldsymbol{T}} = \boldsymbol{\rho}'_{\boldsymbol{S}} \cdot \prod_{i=1}^{\ell} \boldsymbol{\rho}'_{i,\boldsymbol{x}_{\mathsf{inp}(i)}} \cdot \boldsymbol{\rho}'_{\boldsymbol{T}}$ since the value $\boldsymbol{c}$'s are shared for plain and dummy program. It is possible to remove scalar bundlings by dividing $Plain_{\boldsymbol{D}}(\boldsymbol{x})$ by $Dummy_{\boldsymbol{D}}(\boldsymbol{x})$. In other words, we can get $\boldsymbol{d} \cdot \boldsymbol{S}_0 \cdot \prod_{i=1}^{\ell} \boldsymbol{M}_{i,\boldsymbol{x}_{\mathsf{inp}(i)}} \cdot \boldsymbol{T}_0$ for some fixed $\boldsymbol{d}$ from the above division. Since we know all $\boldsymbol{M}$'s, the matrix zeroizing attack works well for the computed quantities.

We remark that the previous analysis [16] analyzed the first candidate *iO* [23]. Whereas the work in [16] heavily relies on the input partitionable property of the single input branching program, our algorithm do not need this property. Moreover, our algorithm can be applied to dual input branching program, so this attack can be applied to wider range of branching programs.

GMMSSZ. Most notable result for BP obfuscation, GMMSSZ, is suggested by Garg *et al.* in TCC 2016 [24]. The authors claim the security of their construction against all known attack. Nevertheless, the matrix zeroizing attack can be applied to their obfuscation.

GMMSSZ obfuscates low-rank matrix branching program, which is evaluated by checking whether the product $\boldsymbol{M}_0 \cdot \prod_{i\in[\ell]} \boldsymbol{M}_{i,\boldsymbol{b}_i} \cdot \boldsymbol{M}_{\ell+1}$ is zero or not. There are two distinctive property of the obfuscation; the uniform random higher dimension embedding and given bookend vectors as inputs. Let $\boldsymbol{M}_0 = (\beta_1, \cdots, \beta_{d_1}), \boldsymbol{M}_{\ell+1} = (\gamma_1, \cdots, \gamma_{d_{\ell+1}})^T$ are the given bookend vectors. The bookend vectors are also extended as $\boldsymbol{H}_0 = (\boldsymbol{M}_0||\boldsymbol{0}), \boldsymbol{H}_{\ell+1} = (\boldsymbol{M}_{\ell+1}||\boldsymbol{U}_{\ell+1})^T$ for randomly chosen $\boldsymbol{U}_{\ell+1}$ in the higher dimension embedding step to remove the higher dimension embedding matrices. Note that the branching programs of this obfuscation are square, we do not restrict the shape of matrices in this section.

For the evaluation, one compute $\widetilde{\boldsymbol{M}}_0 \cdot \prod_{i\in[\ell]} \widetilde{\boldsymbol{M}}_{i,\boldsymbol{b}_i} \cdot \widetilde{\boldsymbol{M}}_{\ell+1}$, which is corresponding to

$$\boldsymbol{D}_{\boldsymbol{S}} \cdot \prod_{i=1}^{\ell} \boldsymbol{D}_{i,\boldsymbol{b}_i} \cdot \boldsymbol{D}_{\boldsymbol{T}} = \boldsymbol{\rho}_{\boldsymbol{S}} \cdot \prod_{i=1}^{\ell} \boldsymbol{\rho}_{i,\boldsymbol{b}_i} \cdot \boldsymbol{\rho}_{\boldsymbol{T}} \cdot \left(\boldsymbol{M}_0 \cdot \prod_{i=1}^{\ell} \boldsymbol{M}_{i,\boldsymbol{b}_i} \cdot \boldsymbol{M}_{\ell+!} \right) \pmod{\langle \boldsymbol{g} \rangle}$$

in $\mathcal{R}$ program by Proposition 5. Since we know all $\boldsymbol{M}$'s, we can compute the ratios of scalar bundlings by

$$\boldsymbol{\rho}_{j,\boldsymbol{b}_j}/\boldsymbol{\rho}_{j,\boldsymbol{b}'_j} = \frac{\boldsymbol{D}_{\boldsymbol{S}} \cdot \prod_{i\in[\ell]} \boldsymbol{D}_{i,\boldsymbol{b}_i} \cdot \boldsymbol{D}_{\boldsymbol{T}} / \boldsymbol{M}_0 \prod_{i\in[\ell]} \boldsymbol{M}_{i,\boldsymbol{b}_i} \cdot \boldsymbol{M}_{\ell+1}}{\boldsymbol{D}_{\boldsymbol{S}} \cdot \prod_{i\in[\ell]} \boldsymbol{D}_{i,\boldsymbol{b}'_i} \cdot \boldsymbol{D}_{\boldsymbol{T}} / \boldsymbol{M}_0 \prod_{i\in[\ell]} \boldsymbol{M}_{i,\boldsymbol{b}'_i} \cdot \boldsymbol{M}_{\ell+1}}$$

for $\boldsymbol{b}, \boldsymbol{b}'$ which are same at all but j-th bit. Therefore, the matrix zeroizing attack well works for the construction of [24]. We remark that this method works for *unknown* bookend matrices with more complicated technique, see Section 6.2.

6.2 Attackable BP Obfuscation, General Case

Now we consider the attackable BP obfuscations in general. We note that an attackable obfuscation without bookends can be considered as the obfuscation with bookends by re-naming the matrices. For example, if we name $\boldsymbol{D}_{\boldsymbol{S}} := \boldsymbol{D}_{1,\boldsymbol{0}} = \boldsymbol{\rho}_{1,\boldsymbol{0}} \cdot \boldsymbol{D}_1$, then we can regard that $\boldsymbol{D}_{\boldsymbol{S}}$ is a left bookend matrix and $\boldsymbol{\rho}_{1,\boldsymbol{0}}$ the corresponding scalar bundling.

The case of obfuscation with bookend matrices is most complex, and requires complicated technique. We will recover the bookend matrices up to constant multiplication, and proceed the algorithm similar to the case of [24].

Recovering the Bookends For the sake of simplicity, we only consider the case of *bookend vectors*. To tackle constructions using bookend matrices, it is suffice to consider a fixed (u, v)-entry of output matrix given in Proposition 2.

If the obfuscation has bookend vectors, then the evaluation of $\mathcal{R}$ program is computed by

$$\boldsymbol{D}_{\boldsymbol{S}} \cdot \prod_{i=1}^{\ell} \boldsymbol{D}_{i,\boldsymbol{b}_i} \cdot \boldsymbol{D}_{\boldsymbol{T}} = \boldsymbol{\rho}_{\boldsymbol{S}} \cdot \prod_{i=1}^{\ell} \boldsymbol{\rho}_{i,\boldsymbol{b}_i} \cdot \boldsymbol{\rho}_{\boldsymbol{T}} \cdot \left(\boldsymbol{S}_0 \cdot \prod_{i=1}^{\ell} \boldsymbol{M}_{i,\boldsymbol{b}_i} \cdot \boldsymbol{T}_0 \right) \pmod{\langle \boldsymbol{g} \rangle}$$

for some vectors $\boldsymbol{S}_0 \in (\mathcal{R}/\langle \boldsymbol{g} \rangle)^{1\times d_1}$ and $\boldsymbol{T}_0 \in (\mathcal{R}/\langle \boldsymbol{g} \rangle)^{d_{\ell+1}\times 1}$. Let $\boldsymbol{S}_0 = (\boldsymbol{\beta}_1, \cdots, \boldsymbol{\beta}_{d_1})$, $\boldsymbol{T}_0 = (\boldsymbol{\gamma}_1, \cdots, \boldsymbol{\gamma}_{d_{\ell+1}})$ and the evaluation $\boldsymbol{D}_{\boldsymbol{S}} \cdot \prod_{i=1}^{\ell} \boldsymbol{D}_{i,\boldsymbol{b}_i} \cdot \boldsymbol{D}_{\boldsymbol{T}}$ is denoted by $Eval_{\boldsymbol{D}}(\boldsymbol{b}_1, \cdots, \boldsymbol{b}_\ell)$.

Our idea is removing $\boldsymbol{\rho}$'s to make equations over $\boldsymbol{S}_0, \boldsymbol{T}_0$. Let $\boldsymbol{b}_{i,t} \in \{0,1\}^w$ for $1 \le i \le \ell$ and $t \in \{0,1\}$ and $\boldsymbol{t} = (t_1, \cdots, t_\ell) \in \{0,1\}^w$. Then the following two values share the same $\boldsymbol{\rho}$'s, precisely $(\boldsymbol{\rho}_{\boldsymbol{S}}\boldsymbol{\rho}_{\boldsymbol{T}})^2 \cdot \prod_{i\in[\ell]} \boldsymbol{\rho}_{i,\boldsymbol{b}_{i,0}} \boldsymbol{\rho}_{i,\boldsymbol{b}_{i,1}}$:

$$Eval_{\boldsymbol{D}}(\boldsymbol{b}_{1,0}, \cdots, \boldsymbol{b}_{\ell,0}) \cdot Eval_{\boldsymbol{D}}(\boldsymbol{b}_{1,1}, \cdots, \boldsymbol{b}_{\ell,1}),$$
$$Eval_{\boldsymbol{D}}(\boldsymbol{b}_{1,t_1}, \cdots, \boldsymbol{b}_{\ell,t_\ell}) \cdot Eval_{\boldsymbol{D}}(\boldsymbol{b}_{1,1-t_1}, \cdots, \boldsymbol{b}_{\ell,1-t_\ell}).$$

We denote $\boldsymbol{S}_0 \cdot \prod_{i=1}^{\ell} \boldsymbol{M}_{i,\boldsymbol{b}_i} \cdot \boldsymbol{T}_0$ by $Eqn_{\boldsymbol{M}}(\boldsymbol{b}_1, \cdots, \boldsymbol{b}_\ell)$. Then, by the above relations, we get a equation for $\boldsymbol{\beta}_1, \cdots, \boldsymbol{\beta}_{d_1}, \boldsymbol{\gamma}_1, \cdots, \boldsymbol{\gamma}_{d_{\ell+1}}$:

$$\begin{aligned} &\frac{Eqn_{\boldsymbol{M}}(\boldsymbol{b}_{1,0}, \cdots, \boldsymbol{b}_{\ell,0}) \cdot Eqn_{\boldsymbol{M}}(\boldsymbol{b}_{1,1}, \cdots, \boldsymbol{b}_{\ell,1})}{Eval_{\boldsymbol{D}}(\boldsymbol{b}_{1,0}, \cdots, \boldsymbol{b}_{\ell,0}) \cdot Eval_{\boldsymbol{D}}(\boldsymbol{b}_{1,1}, \cdots, \boldsymbol{b}_{\ell,1})} \\ &= \frac{Eqn_{\boldsymbol{M}}(\boldsymbol{b}_{1,t_1}, \cdots, \boldsymbol{b}_{\ell,t_\ell}) \cdot Eqn_{\boldsymbol{M}}(\boldsymbol{b}_{1,1-t_1}, \cdots, \boldsymbol{b}_{\ell,1-t_\ell})}{Eval_{\boldsymbol{D}}(\boldsymbol{b}_{1,t_1}, \cdots, \boldsymbol{b}_{\ell,t_\ell}) \cdot Eval_{\boldsymbol{D}}(\boldsymbol{b}_{1,1-t_1}, \cdots, \boldsymbol{b}_{\ell,1-t_\ell})}. \end{aligned}$$

Both side of the equation is homogeneous polynomial of degree 4. If we substitute each degree 4 monomials by another variables, this equation become a homogeneous linear equation of new variables. The number of new variable is $O(d_1^2 d_{\ell+1}^2)$.

Now we assume that we can obtain sufficient number of linearly independent equations generated by the explained way. Then, since the system of linear equations can be solved in $O(M^3)$ time by Gaussian elimination for the number of variable M, we can find all ratios of degree 4 monomials. [7] In other words, we can compute $\boldsymbol{\delta\beta}_1, \cdots, \boldsymbol{\delta\beta}_{d_1}, \boldsymbol{\delta\gamma}_1, \cdots, \boldsymbol{\delta\gamma}_{d_{\ell+1}}$ for some constant $\boldsymbol{\delta}$.

Matrix Zeroizing Attack The remaining part of the attack is exactly same with the attack on GMMSSZ. Precisely, we can recover the ratios of scalar bundlings by computing

$$\boldsymbol{\rho}_{j,\boldsymbol{b}_j} / \boldsymbol{\rho}_{j,\boldsymbol{b}'_j} = \frac{\boldsymbol{D}_{\boldsymbol{S}} \cdot \prod_{i \in [\ell]} \boldsymbol{D}_{i,\boldsymbol{b}_i} \cdot \boldsymbol{D}_{\boldsymbol{T}} / \boldsymbol{S}_0 \prod_{i \in [\ell]} \boldsymbol{M}_{i,\boldsymbol{b}_i} \cdot \boldsymbol{T}_0}{\boldsymbol{D}_{\boldsymbol{S}} \cdot \prod_{i \in [\ell]} \boldsymbol{D}_{i,\boldsymbol{b}'_i} \cdot \boldsymbol{D}_{\boldsymbol{T}} / \boldsymbol{S}_0 \prod_{i \in [\ell]} \boldsymbol{M}_{i,\boldsymbol{b}'_i} \cdot \boldsymbol{T}_0}$$

for $\boldsymbol{b}, \boldsymbol{b}'$ which are same at all but j-th bits. We note that we do not know exact values of $\boldsymbol{S}_0, \boldsymbol{T}_0$, but we recovered $\boldsymbol{\delta S}_0, \boldsymbol{\delta T}_0$ in the above step. Thus we can compute $\boldsymbol{\rho}_{j,\boldsymbol{b}_j} / \boldsymbol{\rho}_{j,\boldsymbol{b}'_j}$ by

$$\frac{\boldsymbol{D}_{\boldsymbol{S}} \cdot \prod_{i \in [\ell]} \boldsymbol{D}_{i,\boldsymbol{b}_i} \cdot \boldsymbol{D}_{\boldsymbol{T}} / (\boldsymbol{\delta S}_0) \prod_{i \in [\ell]} \boldsymbol{M}_{i,\boldsymbol{b}_i} \cdot (\boldsymbol{\delta T}_0)}{\boldsymbol{D}_{\boldsymbol{S}} \cdot \prod_{i \in [\ell]} \boldsymbol{D}_{i,\boldsymbol{b}'_i} \cdot \boldsymbol{D}_{\boldsymbol{T}} / (\boldsymbol{\delta S}_0) \prod_{i \in [\ell]} \boldsymbol{M}_{i,\boldsymbol{b}'_i} \cdot (\boldsymbol{\delta T}_0)}.$$

Therefore the matrix zeroizing attack can be applied to the attackable BP obfuscations, which include all existing BP obfuscations over GGH13.

Acknowledgement

We sincerely thank the anonymous reviewers of Crypto 2018 for their fruitful comments. This work was 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 was based upon work supported by the ARO and DARPA under Contract No.W911NF-15-C-0227.

[7] Here we assume that $\boldsymbol{g}$ is hard to factorize. If $\boldsymbol{g}$ is factorized in the Gaussian elimination procedure, we can proceed the algorithm for a factor of $\boldsymbol{g}$.

References

1. Martin Albrecht, Shi Bai, and Léo Ducas. A subfield lattice attack on overstretched ntru assumptions. In *Annual Cryptology Conference*, pages 153–178. Springer, 2016.
2. Martin R Albrecht, Catalin Cocis, Fabien Laguillaumie, and Adeline Langlois. Implementing candidate graded encoding schemes from ideal lattices. In *Asiacrypt 2015*, volume 9453. Springer, 2015.
3. Prabhanjan Ananth, Divya Gupta, Yuval Ishai, and Amit Sahai. Optimizing obfuscation: Avoiding barrington's theorem. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 646–658. ACM, 2014.
4. Daniel Apon, Nico Döttling, Sanjam Garg, and Pratyay Mukherjee. Cryptanalysis of indistinguishability obfuscations of circuits over ggh13. In *LIPIcs-Leibniz International Proceedings in Informatics*, volume 80. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.
5. Benny Applebaum and Zvika Brakerski. Obfuscating circuits via composite-order graded encoding. In *Theory of Cryptography Conference*, pages 528–556. Springer, 2015.
6. Saikrishna Badrinarayanan, Eric Miles, Amit Sahai, and Mark Zhandry. Post-zeroizing obfuscation: New mathematical tools, and the case of evasive circuits. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 764–791. Springer, 2016.
7. Boaz Barak, Sanjam Garg, Yael Tauman Kalai, Omer Paneth, and Amit Sahai. Protecting obfuscation against algebraic attacks. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 221–238. Springer, 2014.
8. Boaz Barak, Oded Goldreich, Rusell Impagliazzo, Steven Rudich, Amit Sahai, Salil Vadhan, and Ke Yang. On the (im) possibility of obfuscating programs. In *Annual International Cryptology Conference*, pages 1–18. Springer, 2001.
9. Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil Vadhan, and Ke Yang. On the (im) possibility of obfuscating programs. *Journal of the ACM (JACM)*, 59(2):6, 2012.
10. David A Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in nc 1. In *Proceedings of the eighteenth annual ACM symposium on Theory of computing*, pages 1–5. ACM, 1986.
11. Michael Ben-Or and Richard Cleve. Computing algebraic formulas using a constant number of registers. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing*, pages 254–257, 1988.
12. Jean-François Biasse. Subexponential time relations in the class group of large degree number fields. *Adv. in Math. of Comm.*, 8(4):407–425, 2014.
13. Jean-François Biasse, Thomas Espitau, Pierre-Alain Fouque, Alexandre Gélin, and Paul Kirchner. Computing generator in cyclotomic integer rings. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 60–88. Springer, 2017.
14. 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 *Proceedings of the twenty-seventh annual ACM-SIAM symposium on Discrete algorithms*, pages 893–902. SIAM, 2016.

15. Zvika Brakerski and Guy N Rothblum. Virtual black-box obfuscation for all circuits via generic graded encoding. In *Theory of Cryptography Conference*, pages 1–25. Springer, 2014.
16. Yilei Chen, Craig Gentry, and Shai Halevi. Cryptanalyses of candidate branching program obfuscators. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 278–307. Springer, 2017.
17. Jung Hee Cheon, Minki Hhan, and Changmin Lee. Cryptanalysis of the overstretched NTRU problem for general modulus polynomial. *IACR Cryptology ePrint Archive*, 2017:484, 2017.
18. Jung Hee Cheon, Jinhyuck Jeong, and Changmin Lee. An algorithm for NTRU problems and cryptanalysis of the GGH multilinear map without a low-level encoding of zero. *LMS Journal of Computation and Mathematics*, 19(A):255–266, 2016.
19. Jean-Sébastien Coron, Tancrede Lepoint, and Mehdi Tibouchi. Practical multilinear maps over the integers. In *Advances in Cryptology–CRYPTO 2013*, pages 476–493. Springer, 2013.
20. Ronald Cramer, Léo Ducas, Chris Peikert, and Oded Regev. Recovering short generators of principal ideals in cyclotomic rings. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 559–585. Springer, 2016.
21. Nicolas Gama and Phong Nguyen. Predicting lattice reduction. *Advances in Cryptology–EUROCRYPT 2008*, pages 31–51, 2008.
22. Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In *Eurocrypt*, volume 7881, pages 1–17. Springer, 2013.
23. Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *Proceedings of the 2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, pages 40–49. IEEE Computer Society, 2013.
24. 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*, pages 241–268. Springer, 2016.
25. Craig Gentry, Sergey Gorbunov, and Shai Halevi. Graph-induced multilinear maps from lattices. In *Theory of Cryptography*, pages 498–527. Springer, 2015.
26. Paul Kirchner and Pierre-Alain Fouque. Revisiting lattice attacks on overstretched ntru parameters. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 3–26. Springer, 2017.
27. Adeline Langlois, Damien Stehlé, and Ron Steinfeld. Gghlite: More efficient multilinear maps from ideal lattices. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 239–256. Springer, 2014.
28. Kevin Lewi, Alex J Malozemoff, Daniel Apon, Brent Carmer, Adam Foltzer, Daniel Wagner, David W Archer, Dan Boneh, Jonathan Katz, and Mariana Raykova. 5gen: A framework for prototyping applications using multilinear maps and matrix branching programs. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 981–992. ACM, 2016.
29. Fermi Ma and Mark Zhandry. The mmap strikes back: Obfuscation and new multilinear maps immune to clt13 zeroizing attacks. Cryptology ePrint Archive, Report 2017/946, 2017. https://eprint.iacr.org/2017/946.
30. Eric Miles, Amit Sahai, and Mor Weiss. Protecting obfuscation against arithmetic attacks. *IACR Cryptology ePrint Archive*, 2014:878, 2014.

31. Eric Miles, Amit Sahai, and Mark Zhandry. Annihilation attacks for multilinear maps: Cryptanalysis of indistinguishability obfuscation over ggh13. In *Annual Cryptology Conference*, pages 629–658. Springer, 2016.
32. Rafael Pass, Karn Seth, and Sidharth Telang. Indistinguishability obfuscation from semantically-secure multilinear encodings. In *International Cryptology Conference*, pages 500–517. Springer, 2014.
33. Amit Sahai and Mark Zhandry. Obfuscating low-rank matrix branching programs. *IACR Cryptology ePrint Archive*, 2014:773, 2014.
34. Joe Zimmerman. How to obfuscate programs directly. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 439–467. Springer, 2015.

A Extended Attackable BP Obfuscation Model

In this section we introduce an extended model of attackable BP obfuscation by our attack. The extended attackable BP obfuscation is modified in the randomization step to embraces the obfuscation in [15]. The definition of extended attackable conditions for randomization is as follows, which is similar to Definition 3:

Definition 6 (Extended Attackable Conditions for Randomization). *For a branching program $P = \left\{\boldsymbol{M}_{i,\boldsymbol{b}} \in \mathbb{Z}^{d_i \times d_{i+1}}\right\}_{i\in[\ell],\boldsymbol{b}\in\{0,1\}^w}$, the extended attackable randomized branching program is the set*

$$\begin{aligned} Rand(P) = &\left\{\boldsymbol{R}_{i,\boldsymbol{b}}, \boldsymbol{R}'_{i,\boldsymbol{b}} \in \mathbb{Z}^{d_i \times d_{i+1}}\right\}_{i\in[\ell],\boldsymbol{b}\in\{0,1\}^w} \\ &\cup \left\{\boldsymbol{R}_{\boldsymbol{S}}, \boldsymbol{R}'_{\boldsymbol{S}} \in \mathbb{Z}^{d_0 \times d_1}, \boldsymbol{R}_{\boldsymbol{T}}, \boldsymbol{R}'_{\boldsymbol{T}} \in \mathbb{Z}^{d_{\ell+1} \times d_{\ell+2}}\right\} \\ &\cup \left\{\mathsf{aux}_{J,\boldsymbol{b}}, \mathsf{aux}'_{J,\boldsymbol{b}}\right\}_{J\subset[N],\boldsymbol{b}\in\{0,1\}^{w\times|J|}} \end{aligned}$$

satisfying the following properties, where $d_0, d_{\ell+2}, e_i$'s are integers.
1. There exist matrices $\boldsymbol{S}_0, \boldsymbol{S}'_0 \in \mathbb{Z}^{d_0\times d_1}, \boldsymbol{T}_0, \boldsymbol{T}'_0 \in \mathbb{Z}^{d_\ell \times d_{\ell+1}}$ and scalars $\boldsymbol{\alpha}_{\boldsymbol{S}}, \boldsymbol{\alpha}'_{\boldsymbol{S}}, \boldsymbol{\alpha}_{\boldsymbol{T}}, \boldsymbol{\alpha}'_{\boldsymbol{T}}$, $\{\boldsymbol{\alpha}_{i,\boldsymbol{b}}, \boldsymbol{\alpha}'_{i,\boldsymbol{b}}\}_{i\in[\ell],\boldsymbol{b}\in\{0,1\}^w}$ such that the following equations hold for all $\{\boldsymbol{b}_i \in \{0,1\}^w\}_{i\in[\ell]}$:

$$\begin{aligned} \boldsymbol{R}_S \cdot \prod_{i=1}^{\ell} \boldsymbol{R}_{i,\boldsymbol{b}_i} \cdot \boldsymbol{R}_T &= \boldsymbol{\alpha}_{\boldsymbol{S}} \cdot \prod_{i=1}^{\ell} \boldsymbol{\alpha}_{i,\boldsymbol{b}_i} \cdot \boldsymbol{\alpha}_{\boldsymbol{T}} \cdot \left(\boldsymbol{S}_0 \cdot \prod_{i=1}^{\ell} \boldsymbol{M}_{i,\boldsymbol{b}_i} \cdot \boldsymbol{T}_0\right), \\ \boldsymbol{R}'_S \cdot \prod_{i=1}^{\ell} \boldsymbol{R}'_{i,\boldsymbol{b}_i} \cdot \boldsymbol{R}'_T &= \boldsymbol{\alpha}'_{\boldsymbol{S}} \cdot \prod_{i=1}^{\ell} \boldsymbol{\alpha}'_{i,\boldsymbol{b}_i} \cdot \boldsymbol{\alpha}'_{\boldsymbol{T}} \cdot \left(\boldsymbol{S}'_0 \cdot \prod_{i=1}^{\ell} \boldsymbol{M}'_{i,\boldsymbol{b}_i} \cdot \boldsymbol{T}'_0\right). \end{aligned}$$

2. The evaluation of randomized program is done by checking whether the fixed entries of

$$RP(\boldsymbol{x}) = \prod_{J\subset[N]} \mathsf{aux}_{J,\boldsymbol{x}|_J} \cdot \boldsymbol{R}_{\boldsymbol{S}} \cdot \prod_{i=1}^{\ell} \boldsymbol{R}_{i,\boldsymbol{x}_{\mathsf{inp}(i)}} \cdot \boldsymbol{R}_{\boldsymbol{T}} - \prod_{J\subset[N]} \mathsf{aux'}_{J,\boldsymbol{x}|_J} \cdot \boldsymbol{R}'_{\boldsymbol{S}} \cdot \prod_{i=1}^{\ell} \boldsymbol{R}'_{i,\boldsymbol{x}_{\mathsf{inp}(i)}} \cdot \boldsymbol{R}'_{\boldsymbol{T}}$$

is zero or not. Especially, there are two integers u, v such that $P(\boldsymbol{x}) = 0 \Rightarrow RP(\boldsymbol{x})[u,v] = 0$.

After randomizing matrices, we encode every entries and scalars of $Rand(P)$ separately by GGH13 multilinear map with respect to the level corresponding to the first index of elements. We denote $\mathsf{enc}(\mathsf{aux}_{J,\boldsymbol{a}})$ by $\widetilde{\mathsf{aux}}_{J,\boldsymbol{a}}$ for each $J \subset [N]$ and $\boldsymbol{a} \in \{0,1\}^{w \times |J|}$.

We note that aux's were not discussed in the main body of our paper. However, our program converting technique is applied with small modification for auxiliary scalars as well. More precisely, for each $\widetilde{\mathsf{aux}}_{J,\boldsymbol{a}}, \widetilde{\mathsf{aux}}_{J,\boldsymbol{b}}$, we compute $\boldsymbol{h} = \widetilde{\mathsf{aux}}_{J,\boldsymbol{a}} / \widetilde{\mathsf{aux}}_{J,\boldsymbol{b}}$ and solve the NTRU problem for the instance $\boldsymbol{h}$. Then we obtain $\boldsymbol{c}_J \cdot (\mathsf{aux}_{J,\boldsymbol{a}} + \boldsymbol{r}_{\boldsymbol{a}} \cdot \boldsymbol{g})$ for small $\boldsymbol{c}_J$. For an auxiliary scalar $\widetilde{\mathsf{aux}}_{J,\boldsymbol{c}}$ corresponding to J, we compute $\boldsymbol{c}_J \cdot (\mathsf{aux}_{J,\boldsymbol{c}} + \boldsymbol{r}_{\boldsymbol{c}} \cdot \boldsymbol{g}) = \boldsymbol{c}_J \cdot (\mathsf{aux}_{J,\boldsymbol{a}} + \boldsymbol{r}_{\boldsymbol{a}} \cdot \boldsymbol{g}) \cdot \widetilde{\mathsf{aux}}_{J,\boldsymbol{c}} / \widetilde{\mathsf{aux}}_{J,\boldsymbol{a}}$. We can recover dummy auxiliaries as well.

From this calculation, $\mathcal{R}$ program is obtained for extended model. the other step such as recovering the ideal $\langle \boldsymbol{g} \rangle$ and the matrix zeroizing attack work correctly as well.

B Examples of Matrix Zeroizing Attack

Obfuscation in [32]. In this section, we prove that obfuscation in [32] cannot be *iO* for general-purpose. This scheme is characterized by several special randomizations; converting to merged branching program which consists of permutation matrices, and choose the right bookend vector $\boldsymbol{T} = \boldsymbol{e}_1$ and no left bookend vector, and then choose identity Kilian matrix $\boldsymbol{K}_0 = \boldsymbol{I}$ at the first left position. It implies that, by Proposition 5, the evaluation of the program is of the form:

$$\prod_{i=1}^{\ell} \boldsymbol{D}_{i,\boldsymbol{b}_i} \cdot \boldsymbol{D}_{\boldsymbol{T}} = \boldsymbol{\rho}_{\boldsymbol{T}} \cdot \prod_{i=1}^{\ell} \boldsymbol{\rho}_{i,\boldsymbol{b}_i} \cdot \prod_{i=1}^{\ell} \boldsymbol{M}_{i,\boldsymbol{b}_i} \cdot \boldsymbol{e}_1 = \boldsymbol{\rho}_{\boldsymbol{T}} \cdot \prod_{i=1}^{\ell} \boldsymbol{\rho}_{i,\boldsymbol{b}_i} \cdot \boldsymbol{e}_k \pmod{\langle \boldsymbol{g} \rangle},$$

where k is an integer computed by $\boldsymbol{M}$'s. Therefore, we can compute $\boldsymbol{\rho}_{\boldsymbol{T}} \cdot \prod_{i=1}^{\ell} \boldsymbol{\rho}_{i,\boldsymbol{b}_i}$ from the computed value. As a next step, we recover ratios of scalar bundlings $\boldsymbol{\rho}_{j,\boldsymbol{b}_j} / \boldsymbol{\rho}_{j,\boldsymbol{b}'_j}$ for $\boldsymbol{b}, \boldsymbol{b}'$ which satisfies $\boldsymbol{b}_i = \boldsymbol{b}'_i$ for all $i \in [\ell]$ except j by computing the ratio $\boldsymbol{\rho}_{\boldsymbol{T}} \cdot \prod_{i=1}^{\ell} \boldsymbol{\rho}_{i,\boldsymbol{b}_i} / \boldsymbol{\rho}_{\boldsymbol{T}} \cdot \prod_{i=1}^{\ell} \boldsymbol{\rho}_{i,\boldsymbol{b}'_i}$. Finally, we can run the matrix zeroizing attack.

Obfuscation in [6]. Badrinarayanan *et al.* suggest a construction for obfuscation based on branching program, especially for *evasive functions* [6].[8]. In this section, we prove that obfuscation of Badrinarayanan *et al.* cannot be a general-purpose *iO*. This construction is for low-rank branching program, thus it do not have dummy matrices and also does not apply higher dimension embeddings.

The original method for their construction is in the bookend; the authors use no bookend matrices and use special form of Kilian randomization at the first

[8] We remark that the construction of [6] is similar to the construction of [33], which is used as a foundation of recent implementation 5Gen [28] and our attack is also applied to [33] in the same manner.

and last matrices. The first and last Kilian matrices are given as follows:

$$\boldsymbol{K}_0 = diag(\beta_1, \cdots, \beta_{d_1}), \boldsymbol{K}_{\ell+1}^{-1} = diag(\gamma_1, \cdots, \gamma_{d_{\ell+1}}),$$

where β_u, γ_v are randomly chosen scalars.

To evaluate the obfuscated program, we see $\left(\prod_{i=1}^{\ell} \widetilde{M}_{i,\boldsymbol{b}_i}\right)[u,v]$ for some u, v. This is corresponding to the following value, which is computed by Proposition 5,

$$\left(\prod_{i\in[\ell]} \boldsymbol{D}_{i,\boldsymbol{b}_i}\right)[u,v] = \beta_u \cdot \gamma_v \cdot \prod_{i\in[\ell]} \boldsymbol{\rho}_{i,\boldsymbol{b}_i} \cdot \left(\prod_{i\in[\ell]} \boldsymbol{M}_{i,\boldsymbol{b}_i}\right)[u,v] \pmod{\langle \boldsymbol{g} \rangle}$$

since $\boldsymbol{S}_0, \boldsymbol{T}_0$ are exactly $\boldsymbol{K}_0, \boldsymbol{K}_{\ell+1}^{-1}$. We then can recover the ratio of scalar bundlings by computing $\prod_{i\in[\ell]} \boldsymbol{D}_{i,\boldsymbol{b}_i}[u,v] / \prod_{i\in[\ell]} \boldsymbol{D}_{i,\boldsymbol{b}'_i}[u,v]$ for $\boldsymbol{b}, \boldsymbol{b}'$ which satisfies $\boldsymbol{b}_i = \boldsymbol{b}'_i$ for all $i \in [\ell]$ except j. Since we computed ratios of scalar bundlings $\boldsymbol{\rho}_{j,\boldsymbol{b}_j} / \boldsymbol{\rho}_{j,\boldsymbol{b}'_j}$, we can run the matrix zeroizing attack.

C Examples of Linear Relationally Inequivalent BPs

We exhibit two examples of two functionally equivalent but linear relationally inequivalent branching programs here. This examples also certify Proposition 3. The first simple example from nondeterministic finite automata is read-once BPs, and the second example comes from Barrington's theorem and thus input-unpartitionable.

C.1 Read-once BPs from NFA

Two read-once BPs in Table 1 are from non-deterministic finite automata and linear relationally inequivalent.

These two BPs are the point function which output 1 only for input 01, but they are linear relationally inequivalent. For example,

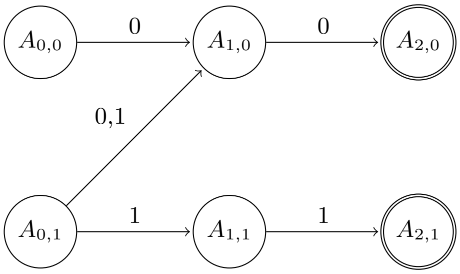
$$\begin{aligned} \boldsymbol{M}_{0,1} \cdot \boldsymbol{M}_{1,0} - \boldsymbol{M}_{0,1} \cdot \boldsymbol{M}_{1,1} &\neq \mathbf{0}, \\ \boldsymbol{N}_{0,1} \cdot \boldsymbol{N}_{1,0} - \boldsymbol{N}_{0,1} \cdot \boldsymbol{N}_{1,1} &= \mathbf{0}. \end{aligned}$$

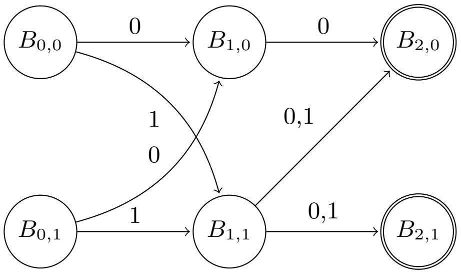
We note that the matrix $\boldsymbol{M}_{i,b}$ is the adjacent matrix between $\{A_{i,c}\}_{c\in\{0,1\}}$ and $\{A_{i+1,c}\}_{c\in\{0,1\}}$, and $\boldsymbol{N}$'s are defined similarly.

C.2 Input-unpartionable BPs from Barrington's Theorem

In the case of Barrington's theorem, the linear relationally inequivalent matrix BPs are more complex. We consider the following two functionally equivalent circuits:

$$\begin{aligned} C_0 &= (X_1 \wedge X_2) \wedge (\neg X_1 \wedge X_3), \\ C_1 &= (\neg X_1 \wedge X_2) \wedge (X_1 \wedge X_3). \end{aligned}$$





$$\boldsymbol{M}_{0,0} = \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix}, \quad \boldsymbol{M}_{1,0} = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix},$$

$$\boldsymbol{M}_{0,1} = \begin{pmatrix} 0 & 0 \\ 1 & 1 \end{pmatrix}, \quad \boldsymbol{M}_{1,1} = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}.$$

$$\boldsymbol{N}_{0,0} = \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix}, \quad \boldsymbol{N}_{1,0} = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix},$$

$$\boldsymbol{N}_{0,1} = \begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix}, \quad \boldsymbol{N}_{1,1} = \begin{pmatrix} 0 & 0 \\ 1 & 1 \end{pmatrix}.$$

Table 1. BPs from NFA

We transform two circuits into the following BPs by Barrington theorem as follow[9]:

$P_{C_0} =$	0:	α_ρ	β_ρ	α_ρ^{-1}	β_ρ^{-1}	e	β_δ	e	β_δ^{-1}	$\cdots$
	1:	e	e	e	e	α_δ	e	α_δ^{-1}	e	$\cdots$
$P_{C_1} =$	0:	e	β_ρ	e	β_ρ^{-1}	α_δ	β_δ	α_δ^{-1}	β_δ^{-1}	$\cdots$
	1:	α_ρ	e	α_ρ^{-1}	e	e	e	e	e	$\cdots$
input bits		1	2	1	2	1	3	1	3	$\cdots$

where τ_σ denotes $\sigma\tau\sigma^{-1}$ for permutations $\tau, \sigma \in S_5$. In the matrix representation, the permutations $\alpha, \beta, \gamma, \rho, \delta$ are of the form

$$\alpha = \begin{bmatrix} 0&1&0&0&0 \\ 0&0&1&0&0 \\ 0&0&0&1&0 \\ 0&0&0&0&1 \\ 1&0&0&0&0 \end{bmatrix}, \beta = \begin{bmatrix} 0&0&1&0&0 \\ 1&0&0&0&0 \\ 0&0&0&0&1 \\ 0&1&0&0&0 \\ 0&0&0&1&0 \end{bmatrix}, \gamma = \begin{bmatrix} 0&0&1&0&0 \\ 0&0&0&0&1 \\ 0&1&0&0&0 \\ 1&0&0&0&0 \\ 0&0&0&1&0 \end{bmatrix}, \rho = \begin{bmatrix} 1&0&0&0&0 \\ 0&0&1&0&0 \\ 0&1&0&0&0 \\ 0&0&0&0&1 \\ 0&0&0&1&0 \end{bmatrix}, \delta = \begin{bmatrix} 1&0&0&0&0 \\ 0&0&0&1&0 \\ 0&0&1&0&0 \\ 0&0&0&0&1 \\ 0&1&0&0&0 \end{bmatrix}.$$

We note that two functionally equivalent branching programs P_{C_0} and P_{C_1} are clearly input-unpartitionable. Now if we consider two (invalid) inputs $\boldsymbol{x} =$

[9] Barrington theorem can be implemented in various ways, but we only consider the first description in [10]. This description also can be found in [4].

0110110111111111 and $\boldsymbol{y} = 1111101011111111$. These yield, for example, $P_{C_0}(\boldsymbol{x}) = \alpha_\rho \cdot e \cdot e \cdot \beta_\rho^{-1} \cdot \alpha_\delta \cdot e \cdot e \cdot e \cdots = \alpha_\rho \cdot \beta_\rho^{-1} \cdot \alpha_\delta = \beta$. The terms in the right $\cdots$ are canceled. Then the equation

$$\begin{gathered} P_{C_0}(\boldsymbol{x}) - P_{C_0}(\boldsymbol{y}) = 0, \\ P_{C_1}(\boldsymbol{x}) - P_{C_1}(\boldsymbol{y}) \neq 0 \end{gathered}$$

hold. Thus two branching programs P_{C_0} and P_{C_1} are functionally equivalent but linear relationally inequivalent.