Cryptanalysis of FRS Obfuscation based on the CLT13 Multilinear Map

Jiseung Kim[⋆] and Changmin Lee[†]

[⋆] Seoul National University, Republic of Korea.
tory154@snu.ac.kr
[†] ENS de Lyon, Laboratoire LIP (U. Lyon, CNRS, ENSL, INRIA, UCBL), France.
changmin.lee@ens-lyon.fr

Abstract. We present classical polynomial time attacks against the FRS branching program obfuscator of Fernando-Rasmussen-Sahai (Asiacrypt'17) (with one zerotest parameter), which is robust against all known classical cryptanalyses on obfuscators, when instantiated with the CLT13 multilinear map.
First of all, we (heuristically) reproduce a new zerotest parameter from the original one. The new zerotest parameter mitigates parameter constraints of the message space recovering algorithm proposed by Coron and Notarnicola (Asiacrypt'19), so it enables us to directly apply the algorithm to the FRS obfuscation.
Then, we propose two cryptanalyses of the FRS obfuscation based on the recovered message space. One analysis enables to obtain all secret elements of CLT13, but it requires extra parameter constraints. On the other hand, the other analysis shows that there exist two functionally equivalent programs such that their obfuscated programs are computationally distinguishable. Thus, the FRS scheme does not satisfy the desired security without any additional constraints.

Keywords: CLT13 multilinear map, FRS obfuscation, indistinguishable obfuscation, input partitionability, zeroizing attack.

1 Introduction

Indistinguishability obfuscation (iO) is a weak notion of the program obfuscation which requires that if two functionally equivalent circuits are given, their obfuscated programs are indistinguishable. The first plausible candidates of iO was proposed by Garg, Gentry, Halevi, Raykova, Sahai and Waters [GGH+13b] using cryptographic multilinear maps. Since then, several candidates of iO have been proposed [AGIS14,MSW14,BR14,PST14,AB15,Zim15,BMSZ16,GMM+16, BD16,HHSSD17,CVW18,DGG+18,BGMZ18] based on the three cryptographic multilinear maps [GGH13a, CLT13, GGH15]. In particular, the CLT13 multilinear map is well known as the most practical scheme used for implementations [LMA+16, CMR17]. The common features of multilinear maps are to be a graded encoding scheme and to provide a zerotest parameter. This parameter distinguishes whether a message of a top-level encoding is zero or not. However, it leads the multilinear map including CLT13 to suffer from several attacks, categorized as zeroizing attacks.

Previous attacks against CLT13 and iO based on CLT13. We briefly review the papers which deal with the cryptanalyses of the CLT13 multilinear map, and iO based on CLT13. Cryptanalyses of the CLT13 multilinear map crucially exploits low level encodings of zero [CHL$^+$15] and almost zero [CN19]. On the other hand, all known cryptanalyses of iO based on CLT13 employ a special property: input zero partitionable.[1]

- Cheon *et al.* [CHL$^+$15] proposed the first cryptanalysis of the CLT13 multilinear map when low level encodings of zero are given. They totally broke the scheme by obtaining all secret elements.
- Coron *et al.* [CGH$^+$15] extended the CHLRS attack [CHL$^+$15] when low level encodings of zeros are not directly given. This implies cryptanalyses of branching program obfuscations over the CLT13 multilinear map when targeted branching programs satisfy input zero partitionable property.
- Coron, Lee, Lepoint, Tibouchi [CLLT17] extended a [CGH$^+$15] attack using the vectorization identity under the obfuscation of branching program also satisfying input zero partitionable.
- Coron and Notarnicola [CN19] extended Gentry, Lewko, and Waters's algorithm [GLW14] to recover a message space of CLT13, and presented a way to analyze CLT13 multilinear maps when there is a low level encoding of almost zero, which is a zero vector except one message slot.
  However, given top level encodings of almost zero, the CLT13 multilinear map has been still open although the message space can be recovered.

At Asiacrypt 2017, Fernando, Rasmussen and Sahai [FRS17] proposed a new iO scheme over the CLT13 multilinear map. The FRS construction follows a standard method to construct a branching program obfuscator; 1) randomizing a given branching program (called a randomized program) 2) encoding a randomized branching program using cryptographic multilinear maps (called an obfuscated program). Yet, FRS construction proposed the extra step before the step 1), called FRS conversion. The 'FRS conversion' is a generic conversion from arbitrary branching program BP into an input zero *'unpartitionable'* branching program BP' while preserving functionality. Since the FRS conversion can be used regardless of the randomization step, the FRS conversion allows it to be applied to any iO schemes based on CLT13 multilinear map. Therefore, FRS conversion serves as an important role in thwarting an input zero partition based attacks up to date.

[1] In this paper, we refine the input partitionable property into two concepts; input partitionable and input zero partitionable. Informally, the definition of input zero partitionable requires that the output of branching programs should be the zero. Input partitionability does not have constraints for outputs of branching programs, which is a relax notion of input zero partitionable. See the Definition 1.1 and 1.2 for their definitions.

In summary, justification of the security of iO schemes with FRS conversion remains as the open problem.

1.1 This work

In this paper, we present two cryptanalyses of the FRS obfuscation based on the CLT13 multilinear map.[2] Our attacks consist of three steps: 1) reproduce a new zerotest parameter, 2) recover a message space using Coron and Notarnicola's algorithm with "*two zerotest parameters*", and 3) cryptanalyze the FRS obfuscation in two ways.

Both analyses require a message space of original branching program, but the message space recovering algorithm proposed by Coron and Notarnicola [CN19] is not applicable to the FRS obfuscation with the zerotest parameter because of the parameter issue. To bridge the gap, we reproduce a new zerotest parameter, which relaxes parameter constraints of a message space recovering algorithm. We are then able to show by using the recovered message space in two perspectives that the FRS obfuscation does not have the desired security.

As an implication, a new attack shows that combining FRS conversion with any iO schemes based on the CLT13 multilinear map does not improve the security of these schemes. More precisely, our contributions are as follows:

1. **Reproduce another zerotest parameter.** Our first main contribution is to (heuristically) make a new zerotest parameter by using the given zerotest parameter.

To formally describe the results, we introduce some parameters.

- α: the bit-length of message space of CLT13
- η: the bit-length of secret primes of CLT13
- n: the number of secret primes of CLT13
- β: the bit-size used in a zerotest parameter of CLT13.
- ν: the bit-size of extraction bits of CLT13
- θ: the number of non-zero plaintext slots of CLT13
- k: the number of CLT13 encodings
- ι: the root Hermite constant of employed lattice reduction algorithm

Then, the algorithm proposed by Coron and Notarnicola [CN19] with a zerotest parameter recovers a message space as long as

$$\alpha\theta(1 + 1/k) + \iota(k + 1) < \nu.$$

Additionally, this condition can be optimized as

$$\alpha\theta + 2\sqrt{\alpha\theta\iota} + \iota < \nu$$

[2] Throughout this paper, we consider the FRS obfuscation with one zerotest parameter. It is an usual construction of this field.

with $k = \sqrt{\alpha\theta/\iota}$.

Unfortunately, in the FRS obfuscation constraints, α has large size because the scheme performs the evaluation of branching program in a composite modulus of the product n primes. Therefore there is no guarantee that the above condition is always met. Hence, the recovering algorithm proposed by the paper [CN19] cannot be used directly if the obfuscation has one zerotest parameter

On the other hand, when two zerotest parameters are available, the bounds improve to $2\sqrt{\alpha\iota} + \alpha/2 + \iota < \nu$ [CN19, Sec. 4.3] and it is always satisfied with the parameter constraints $\nu \geq \alpha + \beta + 5$ [CLT13, Lem. 3]. Therefore we require at least two zero testing parameters to be applied to the Coron *et al*'s algorithm to the FRS obfuscation.

Consequently, we relax parameter constraints of recovering a message space for the FRS scheme. To achieve the goal, we consider an algorithm which reproduces one additional zerotest parameter based on the Cheon *et al*'s approach [CCH+19]. Similar to the paper, we show that any short vector of some lattice generated by CLT13 encodings leads to get a new zerotest parameter. We use the given zerotest parameter to find a short vector of the desired size. Then a new zerotest parameter can be reproduced in polynomial time in n, η, β, α under mild assumptions when it holds that

$$k > \frac{n(\eta - \beta)}{\alpha + \beta + \lambda},$$

where λ is the security parameter. For more details, please see Section 4.1.

2. **Nullifying the FRS conversion.** As the second contribution, we present our main technique to nullify the FRS conversion. As mentioned above, Coron *et al.* [CLLT17] suggested a polynomial time attack when 'input zero partitionability' holds for given branching programs. The essential way to prevent the attack used in the FRS obfuscation is to perform encoding between the original branching program and input zero 'unpartitionability' functions in parallel over the CLT13 multilinear map.

We aim at annihilating the effect of input zero unpartitionability functions. In other words, this technique leads to obtain evaluations of a randomized program over plaintext modulus G_1 from evaluations of the obfuscated program defined over encoding modulus N.

2-1. **Recovering all secret primes of the FRS obfuscation.** Using reproducing and the nullifying technique, we describe how to recover all secret primes of CLT13 multilinear map when top level encodings of almost zero and two zerotest parameters are given.

As a consequence, it allows us to decrypt the CLT13 multilinear map encodings of the FRS scheme, and shows that FRS obfuscation does not satisfy the desired security notion.

Let G_1 be an α-bit modulus integer of the recovered message space and p_1 the one of secret prime factors of N used in the CLT13 multilinear map. Then,

$[p_{zt}/p'_{zt}]_{p_1} = [h/h']_{p_1}$ is satisfied for two zero testing parameters p_{zt} and p'_{zt} and β-bit integers h and h'. Our approach is to compute $[h/h']_{G_1}$ from the nullified program described in the second contribution. Therefore the condition $\beta < \alpha/2$ makes it clearly possible to recover h and h' from the ratio. Then they can be used to recover p_1 as well. Through the similar process, all prime factors of N can be recovered.

Compared to the way of recovering all secret elements using the intermediate level encodings of almost zero presented in the Coron and Notarnicola's paper [CN19, Sec.5], our algorithm has the advantage of requiring only 'top level encodings of almost zero', even though a new condition $\beta < \alpha/2$ is needed.

2-2. **FRS attack with independence on β.** As the last contribution, we show that the FRS obfuscation over the CLT13 multilinear map is not secure regardless of β. In other words, there exist two functionally equivalent branching programs $\boldsymbol{P}$ and $\boldsymbol{Q}$ such that for given $\boldsymbol{P}$ and $\boldsymbol{Q}$ and an obfuscated program of one of them, one can distinguish which one is obfuscated in polynomial time.

By the nullification in the second contribution, we can get a randomized program defined over G_1. As a next step, we propose an algorithm to analyze the randomized program defined over G_1. Hence, we can determine which one branching program is obfuscated although we cannot fully recover elements used in the randomization step.

Furthermore, our attack has an advantage of the class of attackable branching program compared to previous works. As mentioned before, previous attacks employed a branching program obfuscation via input 'zero' partitionability, which is formally defined as follows:

Definition 1.1 (Input 'zero' partitionablility, [FRS17]) *Let $\boldsymbol{v}$ be a vector in $\mathbb{N}^t$ and $f : \mathbb{Z}_{\boldsymbol{v}} \to \{0, \perp\}$ be a function. An input zero partition for f of degree k is a tuple*

$$\mathcal{I}_f^k = (\sigma \in S_t, \{a_i\}_{i\in[k]} \subset \mathbb{Z}_{\boldsymbol{u}_1}, \{c_j\}_{j\in[k]} \subset \mathbb{Z}_{\boldsymbol{u}_2})$$

satisfying $a_i \neq a_j$ and $c_i \neq c_j$ for all $i, j \in [k]$ with $i \neq j$ and $\sigma(\boldsymbol{u}_1||\boldsymbol{u}_2) = \boldsymbol{v}$ such that for all $i, j \in [k]$, $f(\sigma(a_i||c_j))$ should be 'the zero'.
If any PPT adversary cannot find a tuple $\mathcal{I}_f^k$, we say f is input zero unpartitionable.

On the other hand, we extend a target class to branching programs having input partitionablility that is relaxation of the definition of input zero partitionable;

Definition 1.2 (Input partitionability) *For a vector $\boldsymbol{v} \in \mathbb{N}^t$, a function $f : \mathbb{Z}_{\boldsymbol{v}} \to \{0, \perp\}$ is input partitionable of degree k if there exists a tuple*

$$\mathcal{I}_f^k = (\sigma \in S_t, \{a_i\}_{i\in[k]} \subset \mathbb{Z}_{\boldsymbol{u}_1}, \{c_j\}_{j\in[k]} \subset \mathbb{Z}_{\boldsymbol{u}_2})$$

satisfying $a_i \neq a_j$ and $c_i \neq c_j$ for all $i, j \in [k]$ with $i \neq j$ and $\sigma(\boldsymbol{u}_1||\boldsymbol{u}_2) = \boldsymbol{v}$.
If any PPT adversary cannot find a tuple $\mathcal{I}_f^k$, we say f is input unpartitionable.

We remark that since the output of branching program does not have to be zero unlike input zero partitionability, any single input branching program is always converted into input partitionable branching program using the vectorization identity

$$\mathsf{vec}(A \cdot B \cdot C) = (C^T \otimes A) \cdot \mathsf{vec}(B),$$

where vec is an operator from an $m \times n$ matrix into an mn-dimensional column vector obtained by stacking the columns below one another. To show slightly difference between definitions, we give an example of branching programs in Section 5.3, and introduce how our attack works.

Even more, our attack is also applicable to the FRS obfuscated program of multi-input branching programs since any multi-input branching programs are interpreted as single input branching programs when we fix some inputs.

Counter Measure. There exists a simple countermeasure of our attack, which actually prohibits recovering a plaintext modulus. As a counter measure, we consider a repeated BPs to increase θ which is set to be 1 in the original FRS obfuscation.

In the FRS obfuscation, the set of branching programs $\mathsf{BP} := \{\mathsf{BP}_1, \cdots, \mathsf{BP}_n\}$ is used once for parallel encoding, but we use it δ times for parallel encoding construction. In other words, we simultaneously encode δ sets of matrices BP. Then, θ, the number of nonzero slots, is at least δ. As a result, the parameter constraints for recovering a plaintext modulus is changed into

$$\alpha\delta + 2\sqrt{\alpha\delta\iota} + \iota < \nu$$

with $k = \sqrt{\alpha\delta/\iota}$ where the parameters are defined as above. Since δ is independent to other parameters, if we set it large, then the above inequality cannot hold. Therefore, one cannot recover a plaintext modulus.

Open Questions. We leave some open problems.

1. The FRS conversion is still applicable to the branching program iO based on composite order GGH13 multilinear map. Can our attack be extended to the case? It is not easy since adversaries require to find a short element of a plaintext space which is an ideal of ring, not enough to recover a plaintext space.
2. Our attack for recovering all secret primes is applicable to cryptanalysis of CLT13 when $\beta < \alpha/2$. However, in CLT13 parameters usually set to $\beta > \alpha$. Is there an attack for CLT13 multilinear when we have several zerotest parameters?

Organization. In Section 2, we introduce preliminaries related to the iO, matrix branching program, tensor product and CLT13 multilinear map. We briefly describe a FRS obfuscation in Section 3, and present our attack through two Sections 4 and 5.

2 Preliminaries

Notations. We use the lower bold letters as vectors, and capital letters as matrices. Sometimes we use 'bold' capital letters to denote matrices. Let $\mathbb{N}$ be the set of natural numbers and $\mathbb{Z}$ the set of integers, respectively. For $n \in \mathbb{N}$, $[n]$ and S_n denote a set of natural numbers $\{1, 2, \cdots, n\}$ and the set of permutations from $[n]$ to $[n]$, respectively. The disjoint union of two sets X and Y are denoted by $X \bigsqcup Y$. For $q \in \mathbb{N}$, we denote $\mathbb{Z}_q$ by the set $\mathbb{Z} \bigcap (-q/2, q/2]$ and use the notation $[x]_q$ to denote the integer in $\mathbb{Z}_q$ congruent to x mod q. Expanding it to a vector $\boldsymbol{v}$, $[\boldsymbol{v}]_q$ is denoted by $([v_i]_q)_i$, where $\boldsymbol{v} = (v_i)_i$.

For distinct primes $p_1, \cdots, p_t$ and integers $x_1, \cdots, x_t$, $\mathsf{CRT}_{(p_1,\cdots,p_t)}(x_1, \cdots, x_t)$ is denoted by the element $m \in \mathbb{Z}_{\prod_i p_i}$ such that $m \equiv x_i \pmod{p_i}$ for all $i \in [n]$. If the list and indices of p_i's and x_i's are clear, we use an abbreviated notation $\mathsf{CRT}_{(p_i)}(x_i)$. The notation $(\boldsymbol{a}||\boldsymbol{b})$ means a concatenation of vectors $\boldsymbol{a}$ and $\boldsymbol{b}$. Similarly. we denote he concatenation of matrices A and B by $[A||B]$ For a vector $\boldsymbol{x} \in \mathbb{N}^n$, we denote by $\mathbb{Z}_{\boldsymbol{x}}$ by the set $\prod_{i=1}^n \mathbb{Z}_{x_i}$ where $\boldsymbol{x} = (x_1, \cdots, x_n)$. For each element $(m_1, \cdots, m_n) \in \prod_{i=1}^n \mathbb{Z}_{x_i}$, the vector can be regarded as an integer $m \in \mathbb{Z}_{\prod_{i=1}^n x_i}$ since $\prod_{i=1}^n \mathbb{Z}_{x_i} \simeq \mathbb{Z}_{\prod_{i=1}^n x_i}$ when x_i's are relatively primes. We sometimes abuse these representations.

To denote a matrix notation, we borrow $(a_{i,j})_{i,j}$ for a matrix whose (i,j)-component is $a_{i,j}$. For two matrices A, B, we denote $\begin{pmatrix} A & \\ & B \end{pmatrix}$ by $\mathsf{diag}(A, B)$. Similarly, $\mathsf{diag}(a_1, \cdots, a_n)$ is denoted by an $n \times n$ matrix whose i-th diagonal entry is a_i, but other entries all zero. Additionally, we denote the $n \times n$ identity matrix by $\boldsymbol{I}_n$.

When given vectors $\{\boldsymbol{v}_i\}_{1 \le i \le n}$, we denote a linear space generated by the vectors over $S \in \{\mathbb{R}, \mathbb{Z}\}$ by $\langle \boldsymbol{v}_1, \ldots, \boldsymbol{v}_n \rangle_S$.

2.1 Matrix Branching Program

A matrix branching program (BP) is the set which consists of an index-to-input function and several matrix chains.

Definition 2.1 *A width w, length h, and a s-ary matrix branching program $\boldsymbol{P}$ over an ℓ-bit input is a set which consists of index-to-input maps $\{\mathsf{inp}_\mu : [h] \to [\ell]\}_{\mu \in [s]}$, sequences of matrices, and two disjoint sets of target matrices*

$$\boldsymbol{P} = \{P_0 \in \mathbb{Z}^{w \times 1}, P_{h+1} \in \mathbb{Z}^{1 \times w}, (\mathsf{inp}_\mu)_{\mu \in [s]}, \{P_{i,\boldsymbol{b}} \in \{0,1\}^{w \times w}\}_{i \in [h], \boldsymbol{b} \in \{0,1\}^s}\}.$$

The evaluation of $\boldsymbol{P}$ on input $\boldsymbol{x} = (x_i)_{i \in [\ell]} \in \{0,1\}^\ell$ is computed by

$$\boldsymbol{P}(\boldsymbol{x}) = \begin{cases} 0 & \text{if } P_0 \cdot \prod_{i=1}^h P_{i,(x_{\mathsf{inp}_\mu(i)})_{\mu \in [s]}} \cdot P_{h+1} = 0 \\ 1 & \text{if } P_0 \cdot \prod_{i=1}^h P_{i,(x_{\mathsf{inp}_\mu(i)})_{\mu \in [s]}} \cdot P_{h+1} \neq 0 \end{cases}.$$

If $s = 1$ and $s = 2$, then they are called a single-input BP and a double-input BP, respectively. Similarly, if $s > 3$, it is called a multi-input BP. Remark that any NC^1 circuit can be expressed in the form of the BP using the Barrington's theorem.

2.2 Indistinguishability Obfuscation

Definition 2.2 (Indistinguishability Obfuscation) *A probabilistic polynomial time machine $\mathcal{O}$ is an indistinguishability obfuscator for a circuit class $\mathcal{C} = \{\mathcal{C}_\lambda\}$ if the following conditions are satisfied*

- *For all security parameters $\lambda \in \mathbb{N}$, for all circuits $C \in \mathcal{C}_\lambda$, for all inputs $\boldsymbol{x}$, the following probability holds:*
$$\Pr\left[C'(\boldsymbol{x}) = C(\boldsymbol{x}) : C' \leftarrow \mathcal{O}(\lambda, C)\right] = 1.$$
- *For any p.p.t distinguisher 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, $C_1 \in \mathcal{C}_\lambda$, $C_0(\boldsymbol{x}) = C_1(\boldsymbol{x})$ for all inputs $\boldsymbol{x}$ implies*
$$\left|\Pr\left[D(\mathcal{O}(\lambda, C_0)) = 1\right] - \Pr\left[D(\mathcal{O}(\lambda, C_1)) = 1\right]\right| \leq \alpha(\lambda).$$

Generally, direct approach of the constructing iO mainly consists of three parts: 1) Convert the circuits to matrix branching programs, 2) Randomize these matrices, and 3) Obfuscate them using cryptographic multilinear maps [GGH13a, CLT13, GGH15].

2.3 Tensor product and vectorization

For any two matrices $A = (a_{ij})_{i,j} \in \mathbb{Z}^{m\times n}$ and $B \in \mathbb{Z}^{p\times q}$, a tensor product of matrices $A \otimes B$ is defined as a $mp \times nq$ integer matrix such that

$$A \otimes B := \begin{pmatrix} a_{11}\cdot B & \cdots & a_{1m}\cdot B \\ \vdots & \ddots & \vdots \\ a_{n1}\cdot B, & \cdots, & a_{nm}\cdot B \end{pmatrix}.$$

Consider a matrix $C \in \mathbb{Z}^{n\times m}$ whose i-th column is denoted by $\boldsymbol{c}_i$. Then, $\mathsf{vec}(C)$ is a mn-dimensional vector such that

$$\mathsf{vec}(C) = \begin{pmatrix} \boldsymbol{c}_1 \\ \boldsymbol{c}_2 \\ \vdots \\ \boldsymbol{c}_m \end{pmatrix} \in \mathbb{Z}^{mn}.$$

Then, for appropriate matrices A, B and C, the identity holds [Lau05, CLLT17] that
$$\mathsf{vec}(A\cdot B\cdot C) = (C^T \otimes A)\cdot \mathsf{vec}(B).$$

Throughout this paper, we call it 'the vectorization identity'.

2.4 CLT13 multilinear map

Coron *et al.* suggested a candidate of multilinear map over the integers [CLT13]. In this section, we briefly overview the CLT13 multilinear map. Any encodings of the CLT13 multilinear map has the level and one can check whether the message of top-level encoding is zero or not. To see more details, we refer the readers to [CLT13].

In the setting of CLT13, the message and encoding space are $\mathcal{M} = \prod_{i=1}^{n} \mathbb{Z}/\langle G_i \rangle$ and $\mathcal{E} = \mathbb{Z}/\langle \prod_{i=1}^{n} p_i \rangle \simeq \prod_{i=1}^{n} \mathbb{Z}/\langle p_i \rangle$, respectively. Here the integers G_i and the large primes p_i are secret and the $\prod_{i=1}^{n} p_i$ hard to factorize is a public parameter. For the sake of simplicity, we abbreviate the $\prod_{i=1}^{n} p_i$ as N and N/p_i as $\hat{p}_i$ for each i, respectively. An encoding of $\boldsymbol{m} = (m_1, \cdots m_n) \in \mathcal{M}$ at level set $L = \{k\}$ is of the form:

$$\mathsf{enc}_L(\boldsymbol{m}) = \mathsf{CRT}_{(p_j)} \left(\frac{r_j \cdot G_j + m_j}{z_k} \right),$$

where r_j are small integers, and z_k is a secret mask. We simply denote $\mathsf{enc}_k(\boldsymbol{m})$ instead of $\mathsf{enc}_{\{k\}}(\boldsymbol{m})$. To support a κ level multilinearity, κ distinct secret masks $\{z_i\}_{1 \le i \le \kappa}$ are required.

When the summation of two same level encodings or product of two different level encodings has the denominator of small size, we hold $\mathsf{enc}_L(\boldsymbol{m}_1) + \mathsf{enc}_L(\boldsymbol{m}_2) = \mathsf{enc}_L(\boldsymbol{m}_1 + \boldsymbol{m}_2)$ and $\mathsf{enc}_L(\boldsymbol{m}_1) \cdot \mathsf{enc}_{L'}(\boldsymbol{m}_2) = \mathsf{enc}_{L \bigsqcup L'}(\boldsymbol{m}_1 \cdot \boldsymbol{m}_2)$, where the operation of vectors is done component wisely. We simply denote a top level encoding as $\mathsf{enc}_\kappa(\boldsymbol{m})$ at a top level set $[\kappa]$. Additionally the CLT scheme provides a zerotest parameter, which is defined by:

$$p_{zt} = \left[\sum_{j=1}^{n} \left[h_j \cdot \frac{\prod_{k=1}^{\kappa} z_k}{G_j} \right]_{p_j} \cdot \hat{p}_j \right]_N ,$$

where h_j are small integers. We remark that original paper [CLT13] gives several zerotest parameter to check whether the message of encoding is zero vector or not. However, in this work, we only consider iO schemes with one zerotest parameter which is an usual construction. For a top level encoding of zero $\mathsf{enc}_\kappa(\boldsymbol{v}) = \mathsf{CRT}_{(p_j)}(r_j \cdot G_j / \prod_i z_i)$, a zerotest value is defined by product between the top level encoding and a zerotest parameter in modulo N. Then it gives as follow:

$$\begin{aligned} [p_{zt} \cdot \mathsf{enc}_\kappa(\boldsymbol{v})]_N &= \left[\sum_{j=1}^{n} \left[h_j \cdot \frac{\prod_{i=1}^{\kappa} z_i}{G_j} \right]_{p_j} \cdot \hat{p}_j \cdot \left[\frac{r_j \cdot G_j}{\prod_{i=1}^{\kappa} z_i} \right]_{p_j} \right]_N \\ &= \left[\sum_{j=1}^{n} h_j \cdot r_j \cdot \hat{p}_j \right]_N = \sum_{j=1}^{n} h_j \cdot r_j \cdot \hat{p}_j \end{aligned}$$

We note that since the term $r_j \cdot h_j$ is much smaller than p_j, the last equality holds over $\mathbb{Z}$ and the result is also very small compared to N. More precisely, for the bit-size of extraction parameter ν, the size of zerotest result is less than $N \cdot 2^{-\nu-\lambda-2}$ if $\mathsf{enc}_\kappa(\boldsymbol{v})$ is an encoding of zero.

Parameters. We borrow several parameters of CLT13 scheme used to construct the FRS obfuscation. They will be used in Section 5 for introducing the attack. For the security parameter λ, current parameters are set as follows.

- ρ : the bit-size of fresh randomness; satisfy $\rho = \Omega(\lambda)$ to be robust against brute-force attack.
- η : the bit-size of secret primes p_i's; satisfy $\eta = \Omega(\lambda^2)$ for preventing factoring attack for N.
- n : the number of plaintext slots. Namely, $n = \omega(\eta \log \lambda)$
- β : the bit-size of h_i's in the zerotest parameter p_{zt}.
- α : the bit-size of G_i's; takes $\alpha = n \cdot \lambda$.[3]
- ρ_f : the bit-size of maximum randomness at a top level κ; satisfy $\kappa(\mu + \rho + \alpha + 2\log_2 n + 2) + \rho + 1$ with $\mu = \Omega(\lambda)$.
- ν : the bit-size of most significant bits to extract set to $\nu = \eta - \beta - \rho_f - \lambda - 3$. Then, $\nu \geq \alpha + \beta + 5$. (See the Lemma 3, in the [CLT13])

3 Fernando-Rasmussen-Sahai obfuscation

At Asiacrypt 2017, Fernando, Rasmussen and Sahai [FRS17] gave a new iO scheme over CLT13 multilinear map immune to zeroizing attacks. They proposed a general transformation, called 'FRS conversion', by suggesting "stamping functions" for preventing the input zero partition attack. Hence, most of the iO schemes with FRS conversion are robust under the current input zero partition attacks. In this section, we give a high-level description of FRS obfuscation. For a full description, we refer to the paper [FRS17].

First, we borrow a definition of a stamping function H in [FRS17].

Definition 3.1 (Stamping function, [FRS17]) *Let $\boldsymbol{v}_1 \in \mathbb{N}^{t_1}, \boldsymbol{v}_2 \in \mathbb{N}^{t_2}$ be vectors and $\boldsymbol{v}$ denote by the concatenation of $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$. For $F : \mathbb{Z}_{\boldsymbol{v}_1} \to \{0,1\}$ and $H : \mathbb{Z}_{\boldsymbol{v}_1} \to \mathbb{Z}_{\boldsymbol{v}_2}$, construct a function $F' : \mathbb{Z}_{\boldsymbol{v}} \to \{0, \perp\}$ as follows:*

$$F'(\boldsymbol{x}_1 || \boldsymbol{x}_2) = \begin{cases} F(\boldsymbol{x}_1) & \textit{if } H(\boldsymbol{x}_1) = \boldsymbol{x}_2 \\ \perp & \textit{Otherwise.} \end{cases}$$

where $\perp$ symbolizes any nonzero outputs. We say that H secures F if F' is input unpartitionable. This H is called a stamping function.

Note that FRS obfuscation presented three types of initiations for stamping functions. However, we do not consider the concrete stamping functions because

[3] It is a mainly different part from the original parameter constraints of CLT13.

our attack only requires a condition that $H = (H_1 || H_2 || \cdots)$ is the concatenation of independent functions H_i's, which captures current candidates of stamping functions.

We briefly overview how the FRS conversion works. For simplicity, we assume single-input BP and one-to-one function inp. Our goal is to construct a new BP BP' from the original BP and a stamping function H which satisfies

- BP' takes as input of the form $(\boldsymbol{u}||\boldsymbol{v})$, where $\boldsymbol{u}$ is an input of BP.
- checks whether $H(\boldsymbol{u}) = \boldsymbol{v}$; if $H(\boldsymbol{u}) = \boldsymbol{v}$, returns $\mathsf{BP}(\boldsymbol{u})$. Otherwise, it outputs some nonzero values.

Securing a branching program. Suppose that we have a length-ℓ branching program BP over $\{0,1\}^\ell$ and a stamping function $H : \mathbb{Z}_{\boldsymbol{v}_1} \to \mathbb{Z}_{\boldsymbol{v}_2}$ with $\boldsymbol{v}_1 = \{0,1\}^\ell$ and $\boldsymbol{v}_2 \in \mathbb{N}^t$, can be represented by t BPs with the same length $\ell + t$. For a target $\mathsf{BP} = (\{M_{j,b}\}_{j\in[\ell], b\in\{0,1\}}, M_0, M_{\ell+1}, \mathsf{inp})$ with a left (right) bookend vector M_0 $(M_{\ell+1})$, we pad t identity matrices, and redefine a BP and an input function in order to identify the length of $\ell + t$ BPs. We call the new input function inp'.

Thus, we can assume that there are $n := t + 1$ BPs whose lengths are $\ell + t$ denoted by $\{\mathsf{BP}_i\}_{i=1}^n$. For convenience, we ordered BP_1 as the original BP, and others comes from a stamping function H. More formally, let $\mathsf{BP}_i = (\{M_{i,j,c}\}_{j\in[\ell+t], c\in v_j}, M_{i,0}, M_{i,\ell+t+1}, \mathsf{inp}_i)$ where $M_{i,0}$'s ($M_{i,\ell+t+1}$'s) are left (right) bookend vectors, respectively. In order to implement parallel evaluating, we have following constraints about a stamping function H.

- Every BP_i has the same length $\ell + t$ for all $i \in [n]$ and takes inputs from $\mathbb{Z}_{\boldsymbol{v}}$.
- All matrices of BP_i have the same width. If not, we should manipulate the size of matrices by padding the identity matrices.
- For each $i \in [n]$, all matrices and vectors of BP_i are defined over $\mathbb{Z}_{G_i}$ where G_i is the product of n primes $g_{i,j}$.[4] Then, the plaintext space of CLT13 multilinear map is $\prod_{i=1}^n \mathbb{Z}_{G_i}$.
- Every BP_i shares the same input function; for all $2 \le i \le n$, $\mathsf{inp}_i = \mathsf{inp}'$.

Then, a new branching program $\mathsf{BP}' = (\{M'_{j,c}\}_{j\in[\ell+t], c\in v_j}, M'_0, M'_{\ell+t+1}, \mathsf{inp}')$ is defined over the ring $\mathbb{Z}_G := \mathbb{Z}_{\prod_{i=1}^n G_i} \simeq \prod_{i=1}^n \mathbb{Z}_{G_i}$ such that

- $M'_{j,c} \equiv M_{i,j,c} \pmod{G_i}$ for all $i \in [n], j \in [\ell + t]$ and $c \in \mathbb{Z}_{v_j}$. Similarly, we let $M'_0 \equiv M_{i,0} \pmod{G_i}$ and $M'_{\ell+t+1} \equiv M_{i,\ell+t+1} \pmod{G_i}$ for all $i \in [n]$.
- Evaluating BP' at $\boldsymbol{x} \in \mathbb{Z}_{\boldsymbol{v}}$ is the product of
$$M'_0 \times \prod_{j=1}^{\ell+t} M'_{j,x_{\mathsf{inp}'(j)}} \times M'_{\ell+t+1} \pmod{G}.$$

Note that $\mathsf{BP}'(\boldsymbol{x})$ is the zero if and only if $M_{i,0} \cdot \prod_{j=1}^{\ell+t} M_{i,j,x_{\mathsf{inp}'(j)}} \cdot M_{i,\ell+t+1} \pmod{G_i}$ is the zero for all $i \in [n]$.

[4] Here, we assume G_i's are relatively primes.

Next, the branching program is randomized by employing Kilian style randomization and multiplying extra scalars while preserving functionality. We will denote $\widetilde{M}$ by a randomized matrix of M. We defer a description of randomized matrices in Section 5.2.

Last, the randomized matrices are entry-wisely encoded via CLT13 scheme. Note that for each element $m \in \mathbb{Z}_{\prod_{j=1}^{n} G_j}$ in a matrix M, an encoding of m at level set $\{L\}$ is an integer in $\mathbb{Z}_N$ of the form

$$\mathsf{enc}_L(m) \equiv \mathsf{CRT}_{(p_j)}\left(\frac{m_j + G_j r_j}{z_L}\right) \bmod N,$$

where $m_j \equiv m \bmod G_j$, r_j and z_L's are integers derived from CLT13 scheme. As a natural extension, for a matrix $M = (M_{i,j})_{i,j}$ (resp. a vector $M = (M_i)_i$), we denote $\mathsf{enc}_L(M)$ by $(\mathsf{enc}_L(M_{i,j}))_{i,j}$ (resp. $(\mathsf{enc}_L(M_i))_i$).

Let κ be the multilinearity level which is set to $(\ell+t)+2$. Then, any matrices $\widetilde{M'}_{i,c}$ in BP' are encoded as a $\mathsf{enc}_i(\widetilde{M'}_{i,c})$. The matrices M'_0 and $M'_{\ell+t+1}$ can be similarly encoded.

Eventually, a FRS obfuscation scheme outputs

$$\mathcal{O} = \{\{\mathsf{enc}_j(\widetilde{M'}_{j,c})\}_{j\in[\ell+t],c\in v_j}, \mathsf{enc}_0(\widetilde{M'}_0), \mathsf{enc}_{\ell+t+1}(\widetilde{M'}_{\ell+t+1}), \mathsf{inp}', p_{zt}\},$$

where p_{zt} is the zerotest parameter of CLT13. Note that the evaluation on input $\boldsymbol{x}$ of the FRS obfuscation consists of two process. The first process is to compute a product of elements

$$p_{zt} \cdot \mathsf{enc}_0(\widetilde{M'}_0) \times \prod_{j=1}^{\ell+t} \mathsf{enc}_j(\widetilde{M'}_{j,x_{\mathsf{inp}'(j)}}) \times \mathsf{enc}_{\ell+t+1}(\widetilde{M'}_{\ell+t+1}) \pmod N.$$

Throughout the paper, we call the output of the first process pre-evaluation value on $\boldsymbol{x}$. The second one is to check the size of pre-evaluation value. If the size is small, then the obfuscated program outputs the zero. Otherwise, it outputs 1.

4 Recover a plaintext modulus

We describe how to reproduce a new zerotest parameter of CLT13 in Section 4.1, and recall how to recover a plaintext modulus of CLT13 when one has two zerotest parameters in Section 4.2.

The main part of this section is to mitigate attack conditions of [CN19] for recovering a message space $\mathbb{Z}_{G_i}$ by reproducing one more zerotest parameter. More formally, Coron and Notarnicola proved that if parameters (asymptotically) satisfy $\alpha + 2\sqrt{\alpha} < \nu$ with the bit-size of plaintext space α and extraction bit ν, 와then a plaintext space $\mathbb{Z}_{G_1}$ can be recovered. [5] However, the extraction bit ν

[5] The paper [CN19] stated a condition $\alpha < \nu$ when adversary has one zerotest parameter. However, its actual condition is $(1+\epsilon)\alpha < \nu$ for small ϵ. On the other hand, the paper also suggested an attack when multiple zerotest parameters are given, but all iO schemes usually employed only one zerotest parameter.

only require a condition $\alpha \leq \nu$, so we cannot directly use the result of [CN19]. To overcome this gap, we proposed a reproducing technique to provide a new zerotest parameter.

Then, exploiting the two zerotest parameter, anyone can recover the plaintext modulus G_1 in polynomial time under the condition $\sqrt{2\alpha} + \alpha/2 < \nu$. Next we apply the modulus G_1 to cryptanalysis of FRS obfuscation. We describe the attack in Section 5.

4.1 Reproducing a new zerotest parameter from a single zerotest parameter

In this section, we describe how to reproduce a new zerotest parameter from the original zerotest parameter and CLT13 encodings. This approach is similar to the Cheon *et al.*'s algorithm [CCH⁺19].

Recall that the zerotest parameter p_{zt} has the form $\left[\sum_{j=1}^{n}\left[h_j \cdot \frac{\prod_{k=1}^{\kappa} z_k}{G_j}\right]_{p_j} \cdot \hat{p}_j\right]_N$ with $|h_j| \leq 2^{\beta}$ for all $j \in [n]$. Therefore, we focus on reproducing a new quantity of the similar form $d = \left[\sum_{j=1}^{n}\left[d_j \cdot \frac{\prod_{i=1}^{\kappa} z_i}{G_j}\right]_{p_j} \cdot \hat{p}_j\right]_N$ with integers d_j of size $\leq 2^{\beta}$. Then it can also be regarded as a *'zerotest parameter'*.

To obtain a reproduced zerotest parameter, we consider a (column) lattice Λ, which is generated by a matrix $\boldsymbol{B}$. The matrix consists of scaled standard vectors $\{N \cdot \boldsymbol{e}_i\}_{i=1}^{k}$, a k-dimensional vector $[p_{zt} \cdot \boldsymbol{b}]_N = ([p_{zt} \cdot b_j]_N)_j$ where $\boldsymbol{b} = (b_j := \mathsf{enc}_\kappa(\mathbf{0}))$. Note that b_i's are different from each other since the encoding procedure of CLT13 is a probabilistic algorithm. Namely, the matrix $\boldsymbol{B}$ is of the following form:

$$\boldsymbol{B} = \left([p_{zt} \cdot \boldsymbol{b}]_N \parallel N \cdot \boldsymbol{e}_1 \| \cdots \| N \cdot \boldsymbol{e}_k\right) = \begin{pmatrix} [p_{zt} \cdot b_1]_N & N & 0 & \cdots & 0 \\ [p_{zt} \cdot b_2]_N & 0 & N & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ [p_{zt} \cdot b_k]_N & 0 & 0 & \cdots & N \end{pmatrix} \in \mathbb{Z}^{k \times (k+1)}.$$

Before analysis, remark that we only consider the case that the determinant of the lattice is N^{k-1} since the Hermite normal form of the matrix $\boldsymbol{B}$ is a lower triangular matrix whose first diagonal entry is one, and other diagonal entries are all N.[6]

We recall that a zerotest result holds $[p_{zt} \cdot \mathsf{enc}_\kappa(\mathbf{0})]_N \leq N \cdot 2^{-\nu-\lambda-2}$. Then, the size of the first column vector of Λ is relatively smaller than that of the shortest vector guaranteed by Minkowski's theorem, for sufficiently many samples k.

Our goal is to reproduce a new zerotest parameter from a short vector of Λ obtained by running the LLL algorithm on $\boldsymbol{B}$. More specifically, this short vector has of the form $[d \cdot \boldsymbol{b}]_N$, and d will be another zerotest parameter p'_{zt}. The reproducing algorithm involves the following three steps.

[6] If not, we can recover a non-trivial factor of N.

1. Find a short vector $\boldsymbol{g} \in \Lambda$ that is also contained in $\mathbb{R}$-span vector space generated by some n short vectors of Λ, say $\{\boldsymbol{v}_1, \cdots, \boldsymbol{v}_n\}$, by running the LLL algorithm on $\boldsymbol{B}$. Thus, a short vector $\boldsymbol{g}$ can be written as $\sum_{j=1}^{n} d_j \boldsymbol{v}_j$ for some $d_j \in \mathbb{R}$.
2. Show that $d_j \in \mathbb{Z}$ for all $j \in [n]$.
3. Show that the size of such d_j's is smaller than 2^{β} for all $j \in [n]$.

In fact, this approach is true under mild assumptions. From now on, we describe the details step by step.

Claim 1: Find a short vector $\boldsymbol{g}$ of the lattice Λ. First of all, in order to get a short vector $\boldsymbol{g}$ which is linearly independent to the vector $[p_{zt} \cdot \boldsymbol{b}]_N$, we consider running the LLL algorithm on the matrix $\boldsymbol{B}$. The usual LLL algorithm consists of size reduction step and swapping step. The both steps do not increase the size of vector, we can expect to get a vector, which is independent to $[p_{zt} \cdot \boldsymbol{b}]_N$ and smaller than $[p_{zt} \cdot \boldsymbol{b}]_N$, unless the matrix $\boldsymbol{B}$ is already LLL-reduced. Indeed, we heuristically found such a short vector for several parameter settings. So we can state the following Heuristic assumption.

Assumption 1 *The LLL algorithm always outputs at least one vector smaller than* $2^{(n-1)\eta+\beta+\rho_f}$

Hence, the main part of remaining this section is to show a short vector $\boldsymbol{g}$, an output of the LLL algorithm, is of the form $[p'_{zt} \cdot \boldsymbol{b}]_N$, so we can recover p'_{zt}.

In order to do that, we estimate the size of the successive minima $\lambda_i(\Lambda)$. We note that the lattice contains n short vectors of the form $\left[[(\prod_{j=1}^{\kappa} z_j/G_i)]_{p_i} \cdot \hat{p_i} \cdot \boldsymbol{b}\right]_N$ for each $i \in [n]$, called $\boldsymbol{v}_i$. A norm of $\boldsymbol{v}_i$ is bounded by $2^{(n-1)\eta+\rho_f}$ where ρ_f is the maximum bit-size of random 'r' in a κ level encoding.

Then, for sufficiently many samples k, $2^{(n-1)\eta+\rho_f}$ is relatively smaller than $N^{k-1/k}$, which is an upper bound of shortest vector, guaranteed by Minkowski's theorem. So we expect that $\lambda_i(\Lambda) \leq 2^{(n-1)\eta+\rho_f}$ for all $1 \leq i \leq n$. Except these n vectors, we employ the second assumptions about the successive minima $\lambda_i(\Lambda)$ for all $n+1 \leq i \leq k$ of lattice vectors of Λ.

Assumption 2 *For any $n+1 < i \leq k$, i-th successive minima of the lattice Λ is similar to $n+1$-th successive minima $\lambda_{n+1}(\Lambda)$.*

With the above two assumptions, the lower bound of $n+1$-th successive minima $\lambda_{n+1}(\Lambda)$ can be computed. The linearly independent vectors $\boldsymbol{v}_1, \cdots, \boldsymbol{v}_n$ are already contained in the lattice Λ, so i-th successive minima of Λ with $1 \leq i \leq n$ is smaller than an upper bound $2^{(n-1)\cdot\eta+\rho_f}$. Thus, $\prod_{i=1}^{n} \lambda_i(\Lambda)$ is bounded by $2^{(n-1)n\cdot\eta+n\cdot\rho_f}$. Moreover, since $\det \Lambda$ is bounded by $\prod_{i=1}^{k} \lambda_i(\Lambda)$ and the Assumption 2, we observe the followings.

$$N^{k-1}/2^{(n-1)n\cdot\eta+n\rho_f} < \det \Lambda / \left(\prod_{i=1}^{n} \lambda_i(\Lambda) \right)$$

$$\leq \prod_{i=n+1}^{k} \lambda_i(\varLambda) \approx \lambda_{n+1}^{k-n}$$

Thus, we are able to obtain a lower bound of λ_{n+1}, $(N^{k-1}/2^{(n-1)n\cdot\eta+n\rho_f})^{\frac{1}{k-n}}$. According to the bound, a short vector $\boldsymbol{g}$ of the lattice $\varLambda$ is the linear combinations of $\boldsymbol{v}_i$'s, as long as the size of $\boldsymbol{v}_i$'s $1 \leq i \leq n$ is smaller than $\lambda_i(\varLambda)$.

A short vector $\boldsymbol{g}$ in the lattice $\varLambda$ obtained by the LLL algorithm is bounded by $2^{(n-1)\eta+\rho_f+\beta}$ because of the Assumption 1. Therefore, if $2^{(n-1)\eta+\rho_f+\beta}$ is less than the expected $\lambda_{n+1}(\varLambda)$, $\boldsymbol{g}$ and $\{\boldsymbol{v}_i\}_{1\leq i\leq n}$ are included in rank-n space $\langle \boldsymbol{v}'_1, \ldots, \boldsymbol{v}'_n\rangle_{\mathbb{R}}$, where $\boldsymbol{v}'_i$ is a lattice vector of size $\lambda_i(\varLambda)$. In other words, $\boldsymbol{g}$ and $\{\boldsymbol{v}_i\}_{1\leq i\leq n}$ are linear dependent over $\mathbb{R}$, so there exists $d_j \in \mathbb{R}$ such that $\boldsymbol{g} = \sum_{j=1}^{n} d_j \cdot \boldsymbol{v}_j$.

In summary, it is enough to satisfy that the inequality

$$2^{(n-1)\eta+\rho_f+\beta} < \left(N^{k-1}/2^{(n-1)n\cdot\eta+n\rho_f}\right)^{\frac{1}{k-n}}$$

since left-hand side of the inequality is upper bound of a short vector obtained by the LLL algorithm in the lattice $\varLambda$, and right-hand side is lower bound of $\det \varLambda'^{\frac{1}{k-n}}$. Hence, we find a constraint about the number of samples k;

$$k > \frac{n(\eta-\beta)}{\eta-\rho_f-\beta}$$

More simply, it holds that

$$k > \frac{n(\eta-\beta)}{\alpha+\beta+\lambda}$$

since η satisfies $\eta-\rho_f \geq \alpha+2\beta+\lambda+8$. (See the section 2.4.) Consequently, a short vector $\boldsymbol{g}$ is rewritten as $\sum_{j=1}^{n} d_j \cdot \boldsymbol{v}_j$ for some $d_j \in \mathbb{R}$ under the above condition.

Claim 2: Show that $d_j \in \mathbb{Z}$ for all $j \in [n]$. As a second step, we claim that these coefficients d_j's are actually integers. To achieve the goal, we will show that there exists linearly independent vectors $\boldsymbol{v}_{n+1}, \ldots, \boldsymbol{v}_k$ such that $\langle \boldsymbol{v}_1, \ldots, \boldsymbol{v}_k\rangle_{\mathbb{Z}}$ includes any column vectors of the generating matrix $\boldsymbol{B}$ of $\varLambda$. If $\boldsymbol{g}$ is a lattice point of $\varLambda = \langle \boldsymbol{v}_1, \ldots, \boldsymbol{v}_k\rangle_{\mathbb{Z}}$ and is included in $\langle \boldsymbol{v}_1, \ldots, \boldsymbol{v}_n\rangle_{\mathbb{R}}$, then it can then be written as $\sum_{i=1}^{n} d_i \cdot \boldsymbol{v}_i$ with $d_i \in \mathbb{Z}$. Indeed, if $\boldsymbol{g} = \sum_{i=1}^{k} d'_i \boldsymbol{v}_i$ with $d'_i \in \mathbb{Z}$, then it must hold that $d'_i = d_i$ for all $i \in [n]$, and other $d'_i = 0$ for all $n+1 \leq i \leq k$ since coefficients of linear combinations are uniquely determined as $\mathbb{R}$-span vector space.

We first consider a matrix $\boldsymbol{V}$ to find linearly independent vectors $\boldsymbol{v}_{n+1}, \cdots, \boldsymbol{v}_k$.

$$\boldsymbol{V} = [\boldsymbol{v}_1 \| \boldsymbol{v}_2 \| \cdots \| \boldsymbol{v}_n] = \underbrace{\begin{pmatrix} r_{1,1} & r_{2,1} & \cdots & r_{n,1} \\ r_{1,2} & r_{2,2} & \cdots & r_{n,2} \\ \vdots & \vdots & \ddots & \vdots \\ r_{1,k} & r_{2,k} & \cdots & r_{n,k} \end{pmatrix}}_{:=\boldsymbol{R}^T} \cdot \mathsf{diag}(\hat{p}_1, \hat{p}_2, \ldots, \hat{p}_n).$$

It is clear that the first column vector $[p_{zt}\cdot \boldsymbol{b}]_N = \sum_{i=1}^{n} h_i \cdot \boldsymbol{v}_i$ of $\boldsymbol{B}$ is included in $\langle \boldsymbol{v}_1, \ldots, \boldsymbol{v}_n\rangle_{\mathbb{Z}}$. So we aim at adding a new vector to make $N\cdot \boldsymbol{e}_i$ for all $1 \le i \le k$. Now we consider the Hermite normal form of the (row) matrix $\boldsymbol{R}^T$ whose entries can be regarded as random ρ_f-bit integers. If we have $k = O(n)$ samples, there exists a unimodular matrix $\boldsymbol{U} \in \mathbb{Z}^{k\times k}$ such that

$$\boldsymbol{U}\cdot \boldsymbol{R}^T = \begin{pmatrix} \boldsymbol{I}_n \\ \boldsymbol{0}^{(k-n)\times n} \end{pmatrix},$$

where $\boldsymbol{0}^{(k-n)\times n}$ is a $(k-n)\times n$ the zero matrix with overwhelming probability. Let $\hat{\boldsymbol{R}}$ be a matrix of the form

$$\hat{\boldsymbol{R}} = \boldsymbol{U}^{-1}\cdot \begin{pmatrix} \boldsymbol{0}^{n\times(k-n)} \\ \boldsymbol{I}_{k-n} \end{pmatrix}.$$

$\hat{\boldsymbol{R}}$ is also an integer matrix since the unimodular matrix $\boldsymbol{U}^{-1}$ is in $\mathbb{Z}^{k\times k}$. By the construction, a matrix $\left[\boldsymbol{R}^T \| \hat{\boldsymbol{R}}\right]$ is an inverse matrix of $\boldsymbol{U}$ due to $\boldsymbol{U}\cdot\left[\boldsymbol{R}^T \| \hat{\boldsymbol{R}}\right] = \boldsymbol{I}_k$. Moreover, we observe that

$$\begin{aligned} N\cdot \boldsymbol{I}_k &= N\cdot \left[\boldsymbol{R}^T \| \hat{\boldsymbol{R}}\right]\cdot \boldsymbol{U} \\ &= \left[\boldsymbol{R}^T \| \hat{\boldsymbol{R}}\right]\cdot \underbrace{\mathsf{diag}(\hat{p}_1,\ldots,\hat{p}_n,N,\ldots,N)\cdot \mathsf{diag}(p_1,\ldots,p_n,1,\ldots,1)}_{N\cdot \boldsymbol{I}_k}\cdot \boldsymbol{U} \\ &= \left[\boldsymbol{V} \| N\cdot\hat{\boldsymbol{R}}\right]\cdot \mathsf{diag}(p_1,\ldots,p_n,1,\ldots,1)\cdot \boldsymbol{U}. \end{aligned}$$

Therefore, if we denote $\boldsymbol{v}_{n+i}$ by i-th column vector of $N\cdot\hat{\boldsymbol{R}}$, then $\langle \boldsymbol{v}_1,\ldots,\boldsymbol{v}_k\rangle_{\mathbb{Z}}$ contains any column vector of $\boldsymbol{B}$ and $\det \Lambda = \det\left[\boldsymbol{V}\| N\cdot\hat{\boldsymbol{R}}\right]$ by the above equation. Hence, the matrix $\left[\boldsymbol{V}\| N\cdot\hat{\boldsymbol{R}}\right]$ is a basis of Λ.

Claim 3: Show that $|d_j| \le 2^{\beta}$ for all $j \in [n]$. The last step is to show that such integers d_j's are sufficiently small. We first observe that

$$\begin{aligned} \boldsymbol{g}^T &= \sum_{j=1}^{n} d_j \boldsymbol{v}_j^T = \Big(\sum_{j=1}^{n} d_j\cdot r_{j,1}\cdot \hat{p_j}, \sum_{j=1}^{n} d_j\cdot r_{j,2}\cdot \hat{p_j}, \cdots, \sum_{j=1}^{n} d_j\cdot r_{j,k}\cdot \hat{p_j}\Big) \\ &= (d_1\cdot\hat{p_1}, d_2\cdot\hat{p_2},\cdots, d_n\cdot\hat{p_n}) \begin{pmatrix} r_{1,1} & r_{1,2} & \cdots & r_{1,k} \\ r_{2,1} & r_{2,2} & \cdots & r_{2,k} \\ \vdots & \vdots & \ddots & \vdots \\ r_{n,1} & r_{n,2} & \cdots & r_{n,k} \end{pmatrix} \\ &=: (d_1\cdot\hat{p_1}, d_2\cdot\hat{p_2},\cdots, d_n\cdot\hat{p_n})\cdot \boldsymbol{R} \end{aligned}$$

We denote i-th row vector of $\boldsymbol{R}$ by $\boldsymbol{r}_i^T$, and a matrix consists of row vectors $\boldsymbol{r}_2^T, \cdots, \boldsymbol{r}_n^T$ by $\boldsymbol{R}'$. We heuristically assume the size of the inner product between $\boldsymbol{r}_1^T$ and $\boldsymbol{r}'$, where $\boldsymbol{r}'$ is an output of the LLL algorithm on the lattice of orthogonal to $\boldsymbol{R}'$ over integers.

Assumption 3 *Let $\boldsymbol{r}'$ be an output vector of the LLL algorithm on the lattice generated by the right-kernel matrix of $\boldsymbol{R}'$. Then, $|\langle \boldsymbol{r}_1, \boldsymbol{r}' \rangle| \approx \|\boldsymbol{r}_1\| \cdot \|\boldsymbol{r}'\|$ where $\boldsymbol{r}_1^T$ is the first row of the matrix $\boldsymbol{R}$ defined as above.*

From the Assumption 3, for all $j \in [n]$, $|d_j|$ is bounded by 2^β since the following approximation holds

$$\begin{aligned} |\langle \boldsymbol{g}, \boldsymbol{r}' \rangle| = |\boldsymbol{g}^T \cdot \boldsymbol{r}'| &= |(d_1 \cdot \hat{p_1}, d_2 \cdot \hat{p_2}, \cdots, d_n \cdot \hat{p_n}) \cdot \boldsymbol{R} \cdot \boldsymbol{r}'| \\ &= |d_1| \cdot |\hat{p_1}| \cdot |\langle \boldsymbol{r}_1, \boldsymbol{r}' \rangle| \approx 2^{(n-1)\eta} \cdot \|\boldsymbol{r}_1\| \cdot \|\boldsymbol{r}'\|. \end{aligned}$$

Thus, we conclude that

$$\begin{aligned} |d_1| &\approx |\langle \boldsymbol{g}, \boldsymbol{r}' \rangle| / \left(2^{(n-1)\eta} \cdot \|\boldsymbol{r}_1\| \cdot \|\boldsymbol{r}'\| \right) \\ &\leq \|\boldsymbol{g}\| \cdot \|\boldsymbol{r}'\| / \left(2^{(n-1)\eta} \cdot \|\boldsymbol{r}_1\| \cdot \|\boldsymbol{r}'\| \right) \\ &= \|\boldsymbol{g}\| / 2^{(n-1)\eta} \cdot \|\boldsymbol{r}_1\| \leq 2^{(n-1)\eta + \beta + \rho_f} / 2^{(n-1)\eta} \cdot \|\boldsymbol{r}_1\| \approx 2^\beta \end{aligned}$$

Since all entries of a matrix $\boldsymbol{R}$ can be regarded as random ρ_f-bit integers, $\boldsymbol{r}_i$ can be used instead of $\boldsymbol{r}_1$. Thus the assumption 3 still holds when for any $2 \leq i \leq n$, and $|d_i|$ is bounded by 2^β for all $i \in [n]$.

Hence, we can reproduce a zerotest parameter from the original zerotest parameter under the above condition with three assumptions.

Experiments We implemented experiments for reproducing a new zerotest parameter using a `SageMath` [Sag19] and a `fpylll` [dt18]. All experiments are performed on an Intel Core i7-6700K CPU at 4.00 GHz. Every parameter n, k, η, ρ_f and β is the same as the previous section. We confirmed that 1) Assumption 1 holds, 2) we reproduce a zerotest parameter. In other words, we verified that for all j, $|d_j|$ derived from the new zerotest parameter $\left[\sum_{j=1}^{n} \left[d_j \cdot \frac{\prod_{i=1}^{\kappa} z_i}{G_j} \right]_{p_j} \cdot \hat{p}_j \right]_N$ is bounded by 2^β, and 3) difference between $|\langle \boldsymbol{r}_1, \boldsymbol{r}' \rangle| \approx \|\boldsymbol{r}_1\| \cdot \|\boldsymbol{r}'\|$ of Assumption 3.

Overall, the **Table 1** shows the parameters we succeeded for experiments 1) and 2). A column 'Diff' means that the bit-size of difference of $\log_2 \|\boldsymbol{r}_1\| + \log_2 \|\boldsymbol{r}'\| - \log_2 |\langle \boldsymbol{r}_1, \boldsymbol{r}' \rangle|$ at each parameter. According to the results, we argue that our algorithm works well in both theoretically and experimentally.

Reproducing a new zerotest from the original one											
n	k	η	ρ_f	β	Diff	n	k	η	ρ_f	β	Diff
20	25	500	30	30	3.69	50	55	500	30	30	2.90
20	30	300	50	50	1.75	50	55	1000	80	80	3.03
20	30	7500	2000	80	3.89	50	75	700	200	50	3.05
20	50	5500	3000	80	2.91	50	85	700	250	80	5.41

Table 1. Experiment results of reproducing a new zerotest parameter.

4.2 Recover a plaintext modulus

In this section, we recall how to apply the algorithm to the FRS obfuscation scheme to recover a message space $\mathbb{Z}_{G_1}$ of an encoded original program when 2-zerotest parameters $p_{zt,1}, p_{zt,2}$ are given. For more details, we refer to the paper [CN19].

Let $\boldsymbol{P} = \{\mathsf{BP}_1, \mathsf{BP}_2, \cdots, \mathsf{BP}_n\}$ be the set of branching programs employed in the FRS obfuscation scheme. Now, we only consider an input $\boldsymbol{x}$ such that $\mathsf{BP}_1(\boldsymbol{x}) = 1$ and $\mathsf{BP}_i(\boldsymbol{x}) = 0$ for all $2 \leq i \leq n$ for obtaining encodings of almost zero.

$\{w_{j,b}\}_{1\leq j\leq k, 1\leq b\leq 2}$ is denoted by the set of pre-evaluation values of $\mathcal{O}(\boldsymbol{P})$ with a zerotest parameter $p_{zt,b}$, on such an input $\boldsymbol{x}$ of the obfuscated program. It will be explained later how to set the number of samples k. By the construction, it can be written as;

$$w_{j,b} = h_{1,b} \cdot (G_1^{-1} \bmod p_1) \cdot \hat{p}_1 \cdot \tilde{m}_{1,j} + \sum_{i=1}^{n} h_{i,b} \cdot r_{i,j} \cdot \hat{p}_i \bmod N,$$

where $r_{i,j}$'s are ρ_f-bit integers and $\tilde{m}_{1,j} \in \mathbb{Z}_{G_1}$ is a non-zero integer. For simplicity, letting $\boldsymbol{w}_u = (w_{j,b})_j^T$, $\zeta_u = h_{1,b} \cdot (G_1^{-1} \bmod p_1) \cdot \hat{p}_1$, $\tilde{\boldsymbol{m}} = (\tilde{m}_{1,j})_j^T$, and $\boldsymbol{r}_b = (\sum_{i=1}^{n} h_{i,b} \cdot r_{i,j} \cdot \hat{p}_i)_j^T$, we observe the following vector equation;

$$\boldsymbol{w}_b = \zeta_b \cdot \tilde{\boldsymbol{m}} + \boldsymbol{r}_b \bmod N \in \mathbb{Z}^k.$$

Note that the size of each vector $\boldsymbol{r}_b$ in the above equation is approximate to the bit-size of $\rho_R = \gamma - \nu$.

Now we consider a lattice

$$\mathcal{L}_1 := \{((B \cdot \boldsymbol{u}_1) \| \boldsymbol{u}_2) \in \mathbb{Z}^{k+2} \mid \langle (\boldsymbol{u}_1 \| \boldsymbol{u}_2), (\boldsymbol{w}_b \| \boldsymbol{e}_b) \rangle \equiv 0 \bmod N \text{ for all } b \in \{1,2\}\},$$

where $\boldsymbol{e}_b \in \mathbb{Z}^2$ is the b-th standard unit vectors and $B = 2^{\rho_R}$ is a scaling factor.

We claim that the lattice $\mathcal{L}_1$ contains k-linearly independent short vectors and these k-short vectors can be used to recover the message space $\mathbb{Z}_{G_1}$.

First of all, in order to show the lattice $\mathcal{L}_1$ includes several short vectors, we consider the following lattice

$$\mathcal{L}_2 := \{\boldsymbol{f} \in \mathbb{Z}^k \mid \langle \boldsymbol{f}, \tilde{\boldsymbol{m}} \rangle \bmod G_1\}.$$

Then, we expect that the lattice $\mathcal{L}_2$ includes k-linearly independent vectors $\{\boldsymbol{f}_j\}_{j=1}^k$ of norms $\leq G_1^{1/k}$ by assuming Gaussian Heuristic on the lattice $\mathcal{L}_2$, since $\det \mathcal{L}_2 = G_1$[7] and $\mathsf{rank}(\mathcal{L}_2) = k$. For these short vectors, let $\langle \boldsymbol{f}_j, \tilde{\boldsymbol{m}} \rangle$ be of the form $c_j \cdot G_1$ for some integer $c_j \in \mathbb{Z}$. Then, we observe that for all $b \in \{0,1\}$,

$$\langle (\boldsymbol{f}_j \| - (\sum_{b=1}^{2} \langle \boldsymbol{f}_j, \boldsymbol{r}_b \rangle + c_j \cdot h_{1,b} \cdot \hat{p}_1) \cdot \boldsymbol{e}_b), (\boldsymbol{w}_b \| \boldsymbol{e}_b) \rangle$$

[7] Here, we assume that $\gcd(\tilde{m}_{11}, \cdots, \tilde{m}_{1k}, G_1) = 1$.

$$
\begin{aligned}
&= \langle \boldsymbol{f}_j, \boldsymbol{w}_b \rangle - \langle \boldsymbol{f}_j, \boldsymbol{r}_b \rangle + c_j \cdot h_{1,b} \cdot \hat{p}_1 \\
&= \langle \boldsymbol{f}_j, \boldsymbol{\zeta}_b \cdot \tilde{\boldsymbol{m}} \rangle + \langle \boldsymbol{f}_j, \boldsymbol{r}_b \rangle - \langle \boldsymbol{f}_j, \boldsymbol{r}_b \rangle - c_j \cdot h_{1,b} \cdot \hat{p}_1 \\
&\equiv 0 \bmod N.
\end{aligned}
$$

It implies that a vector $\hat{\boldsymbol{f}}_j := \Big(B \cdot \boldsymbol{f}_j \| - (\sum_{b=1}^{2} \langle \boldsymbol{f}_j, \boldsymbol{r}_b \rangle + c_j \cdot h_{1,b} \cdot \hat{p}_1) \cdot \boldsymbol{e}_b\Big)$ is in the lattice $\mathcal{L}_1$. In terms of size, this vector contains dominating terms $B \cdot \boldsymbol{f}_j$ and $\langle \boldsymbol{f}_j, \boldsymbol{r}_b \rangle$ of norms $\leq 2^{\rho_R} \cdot \|\boldsymbol{f}_j\|$. In other words, the lattice $\mathcal{L}_1$ contains at least k linearly independent vectors $\hat{\boldsymbol{f}}_j$ of norms (asymptotically) $\leq 2^{\rho_R} \cdot \|\boldsymbol{f}_j\|$.

Then, by applying the LLL algorithm with an approximate factor $2^{k/2}$ to the lattice $\mathcal{L}_1$, we can obtain k linearly independent vectors $\boldsymbol{f}'_j = (B \cdot \boldsymbol{t}_{j,1} \| \boldsymbol{t}_{j,2})$ such that

$$\|\boldsymbol{f}'_j\| \leq 2^{k/2} \cdot 2^{\rho_R} \cdot \|\boldsymbol{f}_j\| \leq 2^{k/2} \cdot 2^{\rho_R} \cdot G_1^{1/k}$$

for $1 \leq j \leq k$.

Next, we show that a lattice generated by $\{\boldsymbol{t}_{j,1}\}_{1 \leq j \leq k}$ has a determinant of G_1 as long as the size of $\boldsymbol{f}'_j$ is upper bounded. To achieve it, we consider another lattice $\mathcal{L}_3$

$$\mathcal{L}_3 := \{(C \cdot u_1 \| \boldsymbol{u}_2) \in \mathbb{Z}^3 \mid \langle (u_1 \| \boldsymbol{u}_2), (\zeta_b \| \boldsymbol{e}_b) \rangle \equiv 0 \bmod N \text{ for all } b \in \{1, 2\}\},$$

where $C = 2^{\rho_R - \alpha}$ is another scaling factor. Then, a lattice $\mathcal{L}_3$ has rank 3 and determinant $C \cdot N^2$. In particular, the lattice $\mathcal{L}_3$ contains a short vector $\boldsymbol{s} = (C \cdot G_1 \| \sum_{b=1}^{2} -h_{1,b} \cdot \hat{p}_1 \cdot \boldsymbol{e}_b)$. In addition, since $\prod_{i=1}^{3} \lambda_i(\mathcal{L}_3)$ is larger than $\det \mathcal{L}_3$, it holds that

$$N^2/G_1 \leq C \cdot N^2/\|\boldsymbol{s}\| \leq \lambda_2(\mathcal{L}_3)^2$$

under the assumption that $\lambda_1(\mathcal{L}_3) = \|\boldsymbol{s}\| \geq C \cdot G_1$ and $\lambda_2(\mathcal{L}_3) \approx \lambda_3(\mathcal{L}_3)$. Therefore, if there exists a lattice point $\boldsymbol{u} \in \mathcal{L}_3$ such that $\|\boldsymbol{u}\| \leq N/G_1^{1/2}$, then $\boldsymbol{u}$ is the same as the shortest vector $\boldsymbol{S}$ up to constant multiples.

On the other hand, for a lattice point $\boldsymbol{f}'_j = (B \cdot \boldsymbol{t}_{j,1} \| \boldsymbol{t}_{j,2}) \in \mathcal{L}_1$ and all $b \in \{1, 2\}$, it holds that

$$
\begin{aligned}
0 &\equiv \langle (\boldsymbol{t}_{j,1} \| \boldsymbol{t}_{j,2}), (\boldsymbol{w}_b \| \boldsymbol{e}_b) \rangle \bmod N \\
&= \zeta_b \cdot \langle \boldsymbol{t}_{j,1}, \tilde{\boldsymbol{m}} \rangle + \langle \boldsymbol{t}_{j,1}, \boldsymbol{r}_b \rangle + \langle \boldsymbol{t}_{j,2}, \boldsymbol{e}_b \rangle \\
&\equiv \langle (\zeta_b \| \boldsymbol{e}_b), (\langle \boldsymbol{t}_{j,1}, \tilde{\boldsymbol{m}} \rangle \| \sum_{i=1}^{2} (\langle \boldsymbol{t}_{j,1}, \boldsymbol{r}_b \rangle + \langle \boldsymbol{t}_{j,2}, \boldsymbol{e}_b \rangle) \cdot \boldsymbol{e}_i) \rangle \bmod N.
\end{aligned}
$$

Thus the lattice $\mathcal{L}_3$ contains a vector

$$\tilde{\boldsymbol{f}'}_j = (\langle \boldsymbol{t}_{j,1}, \tilde{\boldsymbol{m}} \rangle \| \sum_{i=1}^{2} (\langle \boldsymbol{t}_{j,1}, \boldsymbol{r}_b \rangle + \langle \boldsymbol{t}_{j,2}, \boldsymbol{e}_b \rangle) \cdot \boldsymbol{e}_i)$$

derived from $\boldsymbol{f}'_j$. As previously, if the short vector $\tilde{\boldsymbol{f}'}_j$ satisfies an inequality

$$\|\tilde{\boldsymbol{f}'}_j\| \leq (C \cdot N^2/\|\boldsymbol{s}\|)^{1/2} \leq N/G_1^{1/2}, \tag{1}$$

the vector $\tilde{\boldsymbol{f}}'_j$ is the short vector $\boldsymbol{s}$ up to constant. We then hold that $\langle \boldsymbol{t}_{j,1}, \tilde{\boldsymbol{m}} \rangle$ becomes a multiple of G_1. Namely, $\langle \boldsymbol{t}_{j,1}, \tilde{\boldsymbol{m}} \rangle \equiv 0 \bmod G_1$. Since we expect the vectors $\{\boldsymbol{t}_{j,1}\}$ are independent, the set of vectors $\{\boldsymbol{t}_{j,1}\}_{1 \le j \le k}$ of $\boldsymbol{f}'_j$ are basis of $\mathcal{L}_2$. Then, by computing the determinant of $\mathcal{L}_2$, we can recover the integer G_1.

In summary, a short vector of the lattice $\mathcal{L}_1$, satisfying the inequality (1), allows us to recover a basis of the lattice $\mathcal{L}_2$ and G_1. It is clear that each size of $|\langle \boldsymbol{t}_{j,1}, \tilde{\boldsymbol{m}} \rangle|$, $|\langle \boldsymbol{t}_{j,1}, \boldsymbol{r}_b \rangle|$, and $|\langle \boldsymbol{t}_{j,2}, \boldsymbol{e}_b \rangle|$ are bounded by $2^{k/2} \cdot 2^{\rho_R} \cdot G_1^{1/k}$.

Subsequently, $\tilde{\boldsymbol{f}}'_j$ also has the asymptotically same size. Then, the above condition to find a basis of $\mathcal{L}_2$ can be simplified as:

$$\rho_R + \alpha/k + k/2 < \gamma - \alpha/2.$$

Replacing the number of samples k and ρ_R with $\sqrt{2\alpha}$ and $\gamma - \nu$, respectively, gives a concise approximate bound

$$\sqrt{2\alpha} + \alpha/2 < \nu.$$

As a result, we have the following result.

Proposition 4.1 (Adaptation of Proposition 4, [CN19]) *Let $n, \alpha \in \mathbb{N}$ and G_i be distinct α-bit integers for $1 \le i \le n$. Let ν be the number of bits that can be extracted from zerotest in CLT13 multilinear map. Let $d \in \mathbb{N}$ be the number of encodings where the corresponding plaintexts have only one nonzero components modulo the G_i's Assume that*

$$\sqrt{2\alpha} + \alpha/2 < \nu$$

Then, one can recover the plaintext modulus in polynomial time.

By CLT13 parameter condition described in Section 2.4, the condition of proposition 4.1 is always satisfied. Thus, a secret plaintext modulus $\mathbb{Z}_{G_1}$ can be recovered.

5 Cryptanalysis of the FRS obfuscation

In this section, we present two cryptanalyses of the FRS obfuscation. As previously, we assume that the original BP is encoded under the message space $\mathbb{Z}_{G_1}$ in FRS obfuscation and the message space is already recovered.

Suppose there exists two equivalent BPs $\boldsymbol{P}$ and $\boldsymbol{Q}$ and one obfuscated program $\mathcal{O}(\tilde{\boldsymbol{M}})$ for $\boldsymbol{M} = \boldsymbol{P}$ or $\boldsymbol{Q}$. Our goal is to distinguish whether the program $\boldsymbol{M}$ is $\boldsymbol{P}$ or $\boldsymbol{Q}$. The common strategies throughout the section are to nullify the FRS conversion. In other words, we convert an obfuscated program defined on $\mathbb{Z}_N$ into a randomized program defined on $\mathbb{Z}_{G_1}$.

5.1 Recovering prime factors of the FRS obfuscation

The first attack aims at recovering a secret prime factor p_1 of N. If the prime factor p_1 is recovered, it immediately allows us to completely decrypt multilinear map encodings. More precisely, suppose we have two encodings of a randomized matrix at the same level i, which are congruent to $(G_1 \cdot r_1 + \tilde{m}_1)/z_i$ and $(G_1 \cdot r_2 + \tilde{m}_2)/z_i$ in modulo p_1, respectively. We note that the size of each numerator is much smaller than $\sqrt{p_1}$ because we only handle encodings at level i. By applying rational reconstruction in modulo p_1 to a ratio of two encodings $(G_1 \cdot r_1 + \tilde{m}_1)/(G_1 \cdot r_2 + \tilde{m}_2)$, one can recover an integer $G_1 \cdot r_j + \tilde{m}_j$ for each $j \in \{1, 2\}$. Reducing modulo G_1 then reveals exact randomized messages $\tilde{m}_j$.

After then, one can distinguish between the obfuscation of two BPs such that they are inequivalent under randomization process.

Step 1: Nullify the stamping function. As previously stated, we exploit two zerotest parameters $p_{zt,1}$ and $p_{zt,2}$. Let w_1 and w_2 be pre-evaluation values of $\mathcal{O}(\boldsymbol{P})$ on fixed input $\boldsymbol{x}$ such that $\mathsf{BP}_1(\boldsymbol{x}) = 1$ and $\mathsf{BP}_i(\boldsymbol{x}) = 0$ for each $2 \leq i \leq n$, with zerotest parameters $p_{zt,1}$ and $p_{zt,2}$, respectively. Clearly, it can be written as follows

$$w_1 = h_{1,1} \cdot (G_1^{-1} \bmod p_1) \cdot \hat{p}_1 \cdot \tilde{m}_1 + \sum_{i=1}^{n} h_{1,i} \cdot r_i \cdot \hat{p}_i \bmod N$$
$$w_2 = h_{2,1} \cdot (G_1^{-1} \bmod p_1) \cdot \hat{p}_1 \cdot \tilde{m}_1 + \sum_{i=1}^{n} h_{2,i} \cdot r_i \cdot \hat{p}_i \bmod N,$$

for some integers r_i, $\tilde{m}_1$, and $h_{i,j}$.

Next we consider integers $[w_j \cdot G_1]_N$ for each $j \in \{1, 2\}$. As parameter constraints in Sec 2.4, it is exactly of the form $h_{j,1} \cdot \hat{p}_1 \cdot \tilde{m}_1 + \sum_{i=1}^{n} h_{i,1} \cdot r_i \cdot \hat{p}_i \cdot G_1$ for each $j \in \{1, 2\}$. Hence, $[[w_j \cdot G_1]_N]_{G_1}$ outputs a quantity $[h_{j,1} \cdot \hat{p}_1 \cdot \tilde{m}_1]_{G_1}$, which is independent to any stamping function.

Step 2: Find secret primes of N. We now describe how to recover the (secret) prime factor p_1 of N. For $[[w_j \cdot G_1]_N]_{G_1}$ for $j \in \{1, 2\}$, its ratio in modulo G_1 equals to $[h_{1,1}/h_{2,1}]_{G_1}$. Applying to the rational reconstruction algorithm to this value recovers $h_{1,1}$ and $h_{2,1}$ as long as the size of $h_{j,1}$, 2^β, which is smaller than $2^{\alpha/2}$. On the other hand, the ratio $h_{1,1}/h_{2,1}$ is equal to $p_{zt,1}/p_{zt,2}$ in modulo p_1. Hence, we can recover p_1 by computing $\gcd(p_{zt,2} \cdot h_{1,1} - p_{zt,1} \cdot h_{2,1}, N)$.

Similarly, for each $j \in [n]$, let us consider an input$\boldsymbol{v}_j$ such that $\mathsf{BP}(\boldsymbol{v}_j) = 1$ and $\mathsf{BP}(\boldsymbol{v}_i) = 0$ with all $i \neq j$, one can recover each message space moduli G_j and prime factor p_j of N. Overall the attack only uses basic computations, so it runs in a polynomial time.

Remarks. The attack can be applicable to CLT13 when adversary has several encodings of almost zero at the same level when a plaintext modulus G_1 is recovered, and $\beta < \alpha/2$. This attack does not require specific level of encodings such as low level encodings of zero but a parameter β should be smaller than $\alpha/2$

because of rational reconstruction to recover a secret prime. Therefore, for both cases FRS obfuscation and CLT13, the attack can be thwarted by increasing the size of β. In the next section, we describe another attack which is independent to the size of β.

5.2 Cryptanalysis independent to $\boldsymbol{\beta}$

For the second attack, suppose we have two BPs $\boldsymbol{P}$ and $\boldsymbol{Q}$ and one obfuscated program $\mathcal{O}(\boldsymbol{M})$ for $\boldsymbol{M} = \boldsymbol{P}$ or $\boldsymbol{Q}$. Our goal is to distinguish whether the program $\boldsymbol{M}$ is $\boldsymbol{P}$ or $\boldsymbol{Q}$.

The distinguishing algorithm consists of two steps: 1) Nullify the stamping function using the message space, and 2) Determine the obfuscated programs whether $\boldsymbol{M}$ is $\boldsymbol{P}$ or $\boldsymbol{Q}$ as a final step.

Since every BP can be converted into an input partitionable BP, so we assume that we have an input partitionable BP $\boldsymbol{P}$ without loss of generality. Moreover, the program $\boldsymbol{P}$ takes as input $\boldsymbol{x} \in \mathbb{Z}_{\boldsymbol{v}}$ for a vector $\boldsymbol{v} = (v_i)$, BPs $\{\mathsf{BP}_i\}_{2\le i\le t+1}$ from a stamping function H, and the obfuscated program $\mathcal{O}(\boldsymbol{P})$.

For convenience of description, we assume a FRS converted BP $\boldsymbol{P'}$ is given rather than an original program $\boldsymbol{P}$.

$$\begin{aligned}\boldsymbol{P'} &= (\{P'_{j,c}\}_{j\in[t+\ell],c\in v_j}, P'_0, P'_{t+\ell+1}, \mathsf{inp}')\\ \mathcal{O}(\boldsymbol{P}) &= (\{\mathsf{enc}_j(\widetilde{P'}_{j,c})\}_{j\in[t+\ell],c\in v_j}, \mathsf{enc}_0(\widetilde{P'}_0), \mathsf{enc}_{t+\ell+1}(\widetilde{P'}_{t+\ell+1}), \mathsf{inp}'),\end{aligned}$$

where $\widetilde{P'_{j,c}}$ are randomized matrices of $P'_{j,c}$. Note that $P'_{j,c}$ is of the form

$$\mathsf{diag}\left(\alpha_{j,c}K_{j-1}^{-1}\begin{pmatrix}P'_{j,c} & \\ & R_{j,c}\end{pmatrix}K_j, \alpha'_{j,c}K'^{-1}_{j-1}\begin{pmatrix}I & \\ & R'_{j,c}\end{pmatrix}K'_j\right),$$

where $\{\alpha_{j,c}, \alpha'_{j,c}\}$, K_j, K'_j are randomly chosen scalar bundlings and invertible matrices, respectively. Note that there are some constraints on randomly chosen scalars and invertible matrices to preserve its functionality;

$$P'_0 \times \prod_{j=1}^{t+\ell} P'_{j,x_{\mathsf{inp}(j)}} \times P'_{t+\ell+1} = 0 \iff \widetilde{P'}_0 \times \prod_{j=1}^{t+\ell} \widetilde{P'}_{j,x_{\mathsf{inp}(j)}} \times \widetilde{P'}_{t+\ell+1} = 0.$$

Now, we give a technique how a stamping function nullifies in the FRS obfuscation scheme and determine the obfuscated program. More precisely, we describe a relation when $\boldsymbol{P}$ and $\mathcal{O}(\boldsymbol{P})$ are given; if we have $\boldsymbol{P}$ and $\mathcal{O}(\boldsymbol{Q})$, then it may not have any relations.

Step 1: Nullify the stamping function. Suppose we know the plaintext modulus G_1. For a vector $\boldsymbol{x} \in \mathbb{Z}_{\boldsymbol{v}}$, let $w(\boldsymbol{x})$ be the product of

$$\mathsf{enc}_0(\widetilde{P'}_0) \times \prod_{j=1}^{t+\ell} \mathsf{enc}_j(\widetilde{P'}_{j,x_{\mathsf{inp}(j)}}) \times \mathsf{enc}_{t+\ell+1}(\widetilde{P'}_{t+\ell+1}) \pmod N.$$

Then, we observe $w(\boldsymbol{x})$ can be rewritten as $\mathsf{CRT}_{(p_i)}\left(\frac{r_i \cdot G_i + \tilde{m}_i}{\prod_k z_k}\right)$ for some integers r_i, where $\tilde{m}_i = \widetilde{P}_{i,0} \times \prod_{j=1}^{t+\ell} \widetilde{P}_{i,j,x_{\mathsf{inp}(j)}} \times \widetilde{P}_{i,t+\ell+1} \pmod{G_i}$.

Now, we evaluate $p_{zt} \cdot w(\boldsymbol{a}\|\boldsymbol{b}) \pmod N$ whenever $H(\boldsymbol{a}) = \boldsymbol{b}$ and a nonzero $\tilde{m}_1$. Here, each $\tilde{m}_i$ for $2 \le i \le t$ in the evaluation equals to zero from the fact $H(\boldsymbol{a}) = \boldsymbol{b}$. Furthermore, the zerotest value can be regarded as

$$h_1 \cdot \hat{p_1} \cdot (\tilde{m}_1/G_1 \bmod p_1) + \sum_{i=1}^{t} h_i \cdot \hat{p_i} \cdot r_i \pmod N.$$

Multiplying G_1 by the above equation in modulus N, then we have an integer value

$$h_1 \cdot \hat{p}_1 \cdot \tilde{m}_1 + \sum_{i=1}^{t} G_1 \cdot h_i \cdot \hat{p}_i \cdot r_i.$$

if $|G_1 \cdot p_{zt} \cdot w(\boldsymbol{a}\|\boldsymbol{b}) \bmod N| < N/2$. Note that it holds under the current parameter setting; due to $|p_{zt} \cdot w(\boldsymbol{a}\|\boldsymbol{b})| < N \cdot 2^{-\nu-\lambda-2}$ and $\log_2 |G_1| \le \alpha$.

Eventually, by taking modulus G_1 we obtain $h_1 \cdot \hat{p}_1 \cdot \tilde{m}_1 \bmod G_1$, which is only related to evaluation of BP $\boldsymbol{P'}$ at $\boldsymbol{a}$ in $\mathbb{Z}_{G_1}$. In other words, this value does not depend on the value of $\{\mathsf{BP}_i\}_{2 \le i \le t}$ at all.

Step 2: Determine the obfuscated program. Suppose a given BP $\boldsymbol{P'}$ over $\boldsymbol{v} = (\{0,1\}^\ell \| \boldsymbol{v}_2) \in \mathbb{Z}^{\ell+t}$ satisfies a following structure: $\{P'_{j,c} \pmod{G_1}\}_{j,c} = \{P_{1,j,c}\}_{j,c}$ has an input partition. Note that $\{P_{1,j,c}\}_{j,c}$ is defined over $\{0,1\}^\ell$.

More formally, there are partitions P_{X_1}, P_{X_2} and P_{X_3} satisfying

1. $\{P_{1,j,c}\}_{j,c} = P_{X_1} \bigsqcup P_{X_2} \bigsqcup P_{X_3}$
2. $P_{1,j,c} \in P_{X_k}$ for all $j \in X_k$ with respect to $\{0,1\}^\ell = \sigma(X_1 \| X_2 \| X_3)$ with $X_2 = \{0,1\}$ for some permutation $\sigma \in S_\ell$.

Let us denote $W(\boldsymbol{a})$ by $[[G_1 \cdot p_{zt} \cdot w(\boldsymbol{a}\|\boldsymbol{b})]_N]_{G_1}$ for $H(\boldsymbol{a}) = \boldsymbol{b}$. For sufficiently many $\boldsymbol{x}_i \in X_1, \boldsymbol{y}_j \in X_2$, and $\boldsymbol{z}_k \in X_3$, by employing the vectorization identity, we can construct two invertible matrices $\boldsymbol{W}_0$ and $\boldsymbol{W}_1$ such that

$$\boldsymbol{W}_0 = (W(\boldsymbol{x}_i \| 0 \| \boldsymbol{z}_k))_{i,k} = A \cdot B_0 \cdot C$$
$$\boldsymbol{W}_1 = (W(\boldsymbol{x}_i \| 1 \| \boldsymbol{z}_k))_{i,k}. = A \cdot B_1 \cdot C,$$

where A and C are matrices related only to P_{X_1} and P_{X_3}, respectively. Similarly B_0 and B_1 are matrices calculated only by P_{X_2}. Then, the matrix B_b for $b \in \{0,1\}$ can be represented by tensor product of matrices $\{\widetilde{P'}_{1,j,b}\}$, where j is a location of X_2 in $\{0,1\}^\ell$. For simplicity, we denote it as $B_b = \mathcal{A}(\widetilde{P'}_{1,j,b})$ for some function $\mathcal{A}(\cdot)$.[8]

Since a block matrix $\widetilde{P'}_{1,j,c}$ contains a matrix of $P_{1,j,c}$ up to constant multiplications, and the tensor product of block matrices is computed independently for each block, the set of eigenvalues of $B_1 \cdot B_0^{-1}$ contains that of $\mathcal{A}(P_{1,j,1}) \cdot \mathcal{A}(P_{1,j,0})^{-1}$

[8] Note that we know the function $\mathcal{A}(\cdot)$.

up to constant multiplication. Thus, the set of eigenvalues of $\boldsymbol{W}_1 \cdot \boldsymbol{W}_0^{-1}$ also contains the set of eigenvalues of $\mathcal{A}(P_{1,j,1}) \cdot \mathcal{A}(P_{1,j,0})^{-1}$.

Let $\mathcal{E} = \{e_i\}_i$ be the set of eigenvalues of $\boldsymbol{W}_1 \cdot \boldsymbol{W}_0^{-1}$, and $\mathcal{E}' = \{e'_i\}_i$ the set of eigenvalues of $\mathcal{A}(P_{1,j,1}) \cdot \mathcal{A}(P_{1,j,0})^{-1}$. We then consider the set $\mathcal{E}_h := \{\mathcal{E}/e_h\}_{e_h \in \mathcal{E}}$ and $\mathcal{E}'_h := \{\mathcal{E}'/e'_h\}_{e'_h \in \mathcal{E}'}$ which mean we divide all elements in $\mathcal{E}$ and $\mathcal{E}'$ by a fixed e_h and e'_h, respectively. Then, there exists a pair (h, h') such that $\mathcal{E}'_{h'} \subset \mathcal{E}_h$.

Therefore, if any adversary has $\mathcal{O}(\boldsymbol{M})$, $\boldsymbol{P}$ and $\boldsymbol{Q}$, they can determine whether $\boldsymbol{M}$ is $\boldsymbol{P}$ or $\boldsymbol{Q}$ by computing the eigenvalues of $\boldsymbol{W}_1 \cdot \boldsymbol{W}_0^{-1}$, $\mathcal{A}(P_{1,j,1}) \cdot \mathcal{A}(P_{1,j,0})^{-1}$, and $\mathcal{A}(Q_{1,j,1}) \cdot \mathcal{A}(Q_{1,j,0})^{-1}$. Otherwise, $\mathcal{O}(\boldsymbol{Q})$ and $\boldsymbol{P}$ are given, eigenvalues do not have any relations.

Remarks. For the simplicity, we describe our attack on a special BP $\boldsymbol{P}' \pmod{G_1}$ which has three partitions P_{X_1}, P_{X_2} and P_{X_3}. However, it is always possible to repeatedly use the identity about tensor product and vectorization: $\mathsf{vec}(F_1 \cdot F_2 \cdot F_3) = (F_3^T \otimes F_1) \cdot \mathsf{vec}(F_2)$ for some corresponding matrices F_1, F_2 and F_3. (See Section 2.3.) Thus any BP can be regard as BP with input partitionable. The difference seems to be minor, but it is able to extend attackable BP ranges.

As an example, a BP described in the next Section 5.3 is input partitionable, not input zero partitionable, and an obfuscated program of the BP of example has not been cryptanalyzed. However, our attack still works.

We additionally remark that our attack is also applicable to multi-input BPs unlike the previous paper [CLLT17] since every multi-input BPs can be interpreted as single-input BPs when we fix some inputs. For example, if a double-input BP $\{M_{i,b_1,b_2}\}_{b_1,b_2 \in \{0,1\}}$ is given, it can be written as a single input program when we fix b_2 is always the zero for all i and b_1.

Parameters. In the whole attack, there are two parameter constraints. One is the $|G_1 \cdot p_{zt} \cdot w(\boldsymbol{a}\|\boldsymbol{b}) \bmod N| < N/2$, and the other comes from computing a message space $\mathbb{Z}_{G_1}$.

As stated in the preliminaries 2.4, $p_{zt} \cdot w(\boldsymbol{a}\|\boldsymbol{b}) \bmod N$ is less than $N \cdot 2^{-\nu-\lambda-2}$ with the bit-size of output of extraction ν. Moreover, since $\log_2 |G_1| = \alpha$ is usually set to be smaller than ν, $|G_1 \cdot p_{zt} \cdot w(\boldsymbol{a}\|\boldsymbol{b}) \bmod N| < N/2$ always holds for current parameter. (See Section 2.4 and 4.2.)

Complexity. If the identity s times is used to find partitions $\{0,1\}^\ell = \sigma(X_1||X_2||X_3)$ for some permutation $\sigma \in S_\ell$, then the matrix dimension of $\{\boldsymbol{W}_i\}_{i \in \{0,1\}}$ is at most d^{2^s} where d is the dimension of $\mathsf{enc}_j(\widetilde{P'_{j,c}})$ for all j, c. The complexity of whole attack process is dominated by computing eigenvalues of matrix $\boldsymbol{W}_0 \cdot \boldsymbol{W}_1^{-1}$. Therefore, it implies that when the parameter s is fixed as a small integer, it is polynomial time. Compared to the previous attack [CLLT17], the time complexity is the same.

5.3 An example for our attack

In this section, we describe an example which is input partitionable BP, but not input zero partitionable. We first introduce why a BP $\boldsymbol{P}$ is input partitionable. Moreover, we describe how our attack works on this example.

Let us consider a BP $\boldsymbol{P}$ with identity input function inp.

$$\boldsymbol{P} = (\{P_{j,b}\}_{j\in[5],b\in\{0,1\}}, P_0 = (0,1), P_6 = (1,0)^T, \mathsf{inp})$$

$$P_{i,b} = \begin{cases} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \\ \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 2 \\ 1 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \end{cases}$$

We then evaluate the program $\boldsymbol{P}$ at $\boldsymbol{x} = (x_i)_i \in \{0,1\}^5$ as follows:

$$\boldsymbol{P}(\boldsymbol{x}) = \underbrace{P_0 \cdot P_{1,x_1}}_{P'(x_1)} \times \underbrace{P_{2,x_2}}_{P'(x_2)} \times \underbrace{P_{3,x_3}}_{P'(x_3)} \times \underbrace{P_{4,x_4}}_{P'(x_4)} \times \underbrace{P_{5,x_5} \cdot P_6}_{P'(x_5)}.$$

Then, a BP $\boldsymbol{P}$ is not input zero partitionable. More precisely, due to the vectorization identity, an evaluation of $\boldsymbol{P}$ at $\boldsymbol{x}$ as $(P'(x_1) \otimes P'(x_5)) \times (P'(x_2) \otimes I_2) \times \mathsf{vec}(P'(x_3) \cdot P'(x_4))$. We denote it by $M(x_1\|x_5\|x_2\|x_3\|x_4)$. To represent the function M as a matrix multiplication, at least 2^5 elements are required.[9] So if $\boldsymbol{P}$ is input zero partitionable, $M(x_1\|x_5\|x_2\|x_3\|x_4)$ is always the zero for all possible inputs. However, $\boldsymbol{P}$ does not always output the zero, so we cannot construct a matrix with zero outputs Hence, $\boldsymbol{P}$ is 'NOT' input zero partitionable, which obfuscated program is robust against previous attacks.

On the other hand, $\boldsymbol{P}$ always satisfies input partitionable. Moreover, we briefly introduce how our attack works on BP $\boldsymbol{P}$. For a proper order set of $X = Z = \{0,1\}^2$, we can construct two matrices $\boldsymbol{M}_0$ and $\boldsymbol{M}_1$ of the form

$$\boldsymbol{M}_0 = (M(\boldsymbol{x}_i\|0\|\boldsymbol{z}_k))_{i,k} = \begin{pmatrix} 1&0&0&0 \\ 0&1&0&0 \\ 0&0&1&0 \\ 0&0&0&1 \end{pmatrix} \cdot \begin{pmatrix} P_{2,0} & \\ & P_{2,0} \end{pmatrix} \cdot \begin{pmatrix} 2&0&0&1 \\ 0&1&2&0 \\ 0&2&1&0 \\ 1&0&0&2 \end{pmatrix}$$

$$\boldsymbol{M}_1 = (M(\boldsymbol{x}_i\|1\|\boldsymbol{z}_k))_{i,k} = \begin{pmatrix} 1&0&0&0 \\ 0&1&0&0 \\ 0&0&1&0 \\ 0&0&0&1 \end{pmatrix} \cdot \begin{pmatrix} P_{2,1} & \\ & P_{2,1} \end{pmatrix} \cdot \begin{pmatrix} 2&0&0&1 \\ 0&1&2&0 \\ 0&2&1&0 \\ 1&0&0&2 \end{pmatrix}$$

Therefore a matrix $\boldsymbol{M}_1 \cdot \boldsymbol{M}_0^{-1}$ has $\{1\}$ as an eigenvalue. On the other hand, we consider a program $\boldsymbol{Q}$ which is equal to the program $\boldsymbol{P}$ except for $\boldsymbol{Q}_{2,1} = \begin{pmatrix} 2 & 0 \\ 0 & 3 \end{pmatrix}$. Then BPs $\boldsymbol{P}$ and $\boldsymbol{Q}$ have the same functionality. However, in this case, $\{2,3\}$ can be obtained as eigenvalues with the same computation. Hence, eigenvalues of $\boldsymbol{W}_1 \cdot \boldsymbol{W}_0^{-1}$ of an obfuscated program $\mathcal{O}$ can be used to determine which program corresponds to the obfuscated program.

[9] We need two 4×4 matrices.

References

AB15. Benny Applebaum and Zvika Brakerski. Obfuscating circuits via composite-order graded encoding. In *Theory of Cryptography Conference*, pages 528–556. Springer, 2015.

AGIS14. 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.

BD16. Zvika Brakerski and Or Dagmi. Shorter circuit obfuscation in challenging security models. In *International Conference on Security and Cryptography for Networks*, pages 551–570. Springer, 2016.

BGMZ18. James Bartusek, Jiaxin Guan, Fermi Ma, and Mark Zhandry. Return of ggh15: Provable security against zeroizing attacks. In *Theory of Cryptography Conference*, pages 544–574. Springer, 2018.

BMSZ16. 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.

BR14. 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.

CCH$^+$19. Jung Hee Cheon, Wonhee Cho, Minki Hhan, Minsik Kang, Jiseung Kim, and Changmin Lee. Algorithms for crt-variant of approximate greatest common divisor problem. *Number-Theoretic Methods in Cryptology (NutMiC)*, 2019:195, 2019.

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

CHL$^+$15. Jung Hee Cheon, Kyoohyung Han, Changmin Lee, Hansol Ryu, and Damien Stehlé. Cryptanalysis of the multilinear map over the integers. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 3–12. Springer, 2015.

CLLT17. Jean-Sébastien Coron, Moon Sung Lee, Tancrede Lepoint, and Mehdi Tibouchi. Zeroizing attacks on indistinguishability obfuscation over clt13. In *IACR International Workshop on Public Key Cryptography*, pages 41–58. Springer, 2017.

CLT13. 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.

CMR17. Brent Carmer, Alex J Malozemoff, and Mariana Raykova. 5gen-c: multi-input functional encryption and program obfuscation for arithmetic circuits. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 747–764. ACM, 2017.

CN19. Jean-Sébastien Coron and Luca Notarnicola. Cryptanalysis of clt13 multilinear maps with independent slots. 2019.

CVW18. Yilei Chen, Vinod Vaikuntanathan, and Hoeteck Wee. Ggh15 beyond permutation branching programs: Proofs, attacks, and candidates. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology –*

CRYPTO 2018, pages 577–607, Cham, 2018. Springer International Publishing.

DGG$^+$18. Nico Döttling, Sanjam Garg, Divya Gupta, Peihan Miao, and Pratyay Mukherjee. Obfuscation from low noise multilinear maps. In *International Conference on Cryptology in India*, pages 329–352. Springer, 2018.

dt18. The FPyLLL development team. A lattice reduction library. 2018. Available at https://github.com/fplll/fpylll.

FRS17. Rex Fernando, Peter MR Rasmussen, and Amit Sahai. Preventing clt attacks on obfuscation with linear overhead. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 242–271. Springer, 2017.

GGH13a. Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In *Eurocrypt*, volume 7881, pages 1–17. Springer, 2013.

GGH$^+$13b. Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *Proceedings of the 2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, pages 40–49. IEEE Computer Society, 2013.

GGH15. Craig Gentry, Sergey Gorbunov, and Shai Halevi. Graph-induced multilinear maps from lattices. In *Theory of Cryptography*, pages 498–527. Springer, 2015.

GLW14. Craig Gentry, Allison B. Lewko, and Brent Waters. Witness encryption from instance independent assumptions. In *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I*, pages 426–443, 2014.

GMM$^+$16. Sanjam Garg, Eric Miles, Pratyay Mukherjee, Amit Sahai, Akshayaram Srinivasan, and Mark Zhandry. Secure obfuscation in a weak multilinear map model. In *Theory of Cryptography Conference*, pages 241–268. Springer, 2016.

HHSSD17. Shai Halevi, Tzipora Halevi, Victor Shoup, and Noah Stephens-Davidowitz. Implementing bp-obfuscation using graph-induced encoding. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 783–798. ACM, 2017.

Lau05. Alan J Laub. *Matrix analysis for scientists and engineers*, volume 91. Siam, 2005.

LMA$^+$16. 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.

MSW14. Eric Miles, Amit Sahai, and Mor Weiss. Protecting obfuscation against arithmetic attacks. *IACR Cryptology ePrint Archive*, 2014:878, 2014.

PST14. Rafael Pass, Karn Seth, and Sidharth Telang. Indistinguishability obfuscation from semantically-secure multilinear encodings. In *International Cryptology Conference*, pages 500–517. Springer, 2014.

Sag19. Sage Developers. *SageMath, the Sage Mathematics Software System (Version 8.6)*, 2019. `https://www.sagemath.org`.

Zim15. Joe Zimmerman. How to obfuscate programs directly. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 439–467. Springer, 2015.