# Cryptanalysis on the HHSS Obfuscation Arising from Absence of Safeguards

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

***Abstract*—Indistinguishability Obfuscation ($iO$) is a hopeful tool which obfuscates a program with the least-possible leakage, and produces various applications including functional encryption and deniable encryption. Recently, Halevi *et. al.* proposed a state-of-the-art obfuscator implementation, called HHSS obfuscation, in ACM-CCS'17.**

**In this work, we describe a polynomial time distinguishing attack on HHSS obfuscation. In other words, we show that there exist two functionally equivalent branching programs but obfuscated programs are actually distinguishable. This attack implies that HHSS obfuscation fails to achieve a general purpose of $iO$ security. The idea of the attack is quite simple; we multiply a left kernel vector of the branching program $\mathcal{P}$ to an evaluation of obfuscated matrix, which yields a small value when the program $\mathcal{P}$ is obfuscated. Our attack algorithm is also applicable even if evasive functions are obfuscated.**

***Index Terms*—graded encoding scheme, indistinguishability obfuscation**

## I. Introduction

The program obfuscator, also called software obfuscator, is a compiler which takes a program $\mathcal{P}$ as an input, and outputs a new program $\mathcal{O}(\mathcal{P})$ which preserves its functionality but it is infeasible to extract information of the program $\mathcal{P}$ other than natural leakage of program. The most significant of the obfuscation is that the obfuscated program can withstand reverse engineering.

The indistinguishability obfuscation ($iO$), also known as best-possible obfuscation [1], makes a program leak as little information as any other program with the same functionality and a similar size. Equivalently, any adversary cannot distinguish two obfuscated programs which are obfuscation of functionally equivalent program with the same size in a polynomial time. Indeed, it implies that there are many applications of $iO$. When there exists an indistinguishability obfuscation, it realizes functional encryption, deniable encryption, multiparty secure computation, etc [2].

In 2013, Garg *et. al.* first suggested a plausible candidate of $iO$ [3] based on the graded encoding schemes [4]–[6] and a matrix branching program (BP) which represents the program by several matrices and an input function. Subsequently, many variants of Garg *et. al.*'s construction are suggested [7]–[11].

Recently, BP obfuscation over GGH15 graded encoding scheme has been in the spotlight because several schemes claim provable security under the LWE assumption [12], [13]. On top of that, Halevi *et. al.* presented a new construction of $iO$ which was successful in implementation of bfuscatation of read-once branching programs [14].

At the same time, cryptanalyses on obfuscations of general branching programs over three graded encoding schemes [4]–[6] have been suggested [15]–[20]. Thus, there is no $iO$ candidates for general purpose over graded encoding schemes. Instead, the special purpose of $iO$ candidates have still proposed [12], [13], [20]–[22].

Among those candidates, obfuscations of evasive functions have several important applications.[1] For example, the developer employs the obfuscation of the evasive function to update software patches without leaking additional information. (See [21] for more details). In addition, obfuscations of evasive functions can withstand all known polynomial time attack since these attacks employ several encodings of zero from honest evaluations.

**This work.** In this work, we propose a noteworthy polynomial time distinguishing attack on HHSS obfuscation. In other words, we show that there exist two functionally equivalent branching programs but obfuscated programs of them are distinguishable.

In particular, our attack can distinguish between obfuscations of two evasive functions on HHSS obfuscation since it only depends on the left kernel of the matrix branching program.

More specifically, we select one of the two programs which has the short vector belonging to the left kernel. This vector is then multiplied to the matrix obtained by evaluation of the obfuscation. If the given obfuscated program originated from our chosen program, the norm of the output vector is smaller than certain value, otherwise it returns a larger value. Thus, we can find the origin of the given obfuscated program through the size of the newly computed vector.

In addition, we observe variants of HHSS obfuscation are also distinguishable.

- Original HHSS: left kernel attack
- Add scalar bundling without left bookend vector to HHSS: left kernel attack
- Add left bookend vector without scalar bundling to HHSS: in this case, by employing matrix zeroizing attack proposed by Cheon *et. al.* [18].

**Comparison to concurrent and independent work.** Recently, Chen, Vaikuntanathan and Wee proposed another attack

Authors contributed equally to this work. Authors except for C.Lee were with the Department of Mathematical sciences, Seoul National University, Republic of Korea. C.Lee were with the Research Institute of Basic Sciences, Seoul National University, Korea.

[1]Evasive function is a function $f : \{0,1\}^* \to \{0,1\}$ that outputs 0 with overwhelming probability. Some authors use the term to denote functions that output 1 with overwhelming probability. An example of evasive functions is password checker.

on HHSS obfuscation and its variant. The main idea of their attack is to construct a certain form of matrix using several encodings of zero from evaluation and then compute its rank for distinguishing attack. Indeed, their attack, *rank attack*, is applicable to original version of BP obfuscation over GGH15.

The rank attack requires several encodings of zero from honest evaluations, which is hard to get in evasive programs. Therefore, obfuscated evasive programs are secure against the rank attack, whereas our attack is well applied to obfuscated evasive programs by HHSS construction. Our attack is not applicable to GGH15 BP obfuscation with *all safeguards.*

## *A. Overview of the Attack*

We briefly present the GGH15 graded encoding scheme, HHSS obfuscation and our left kernel attack.

**The GGH15 graded encoding scheme.** First, we simply describe the construction of the GGH15 graded encoding scheme. The scheme consists of encoding process and zerotesting procedure.

Let $G = (V, E)$ be a a directed acyclic graph which has a single source node 0 and a single sink node $\ell$. The encoding process takes as input matrices $\mathbf{M}_{i,b} \in \mathbb{Z}^{d\times d}$ under a path $(i-1) \to i$, and outputs the matrices $\mathbf{C}_{i,b} \in \mathbb{Z}_q^{m\times m}$ such that

$$\mathbf{A}_{i-1} \cdot \mathbf{C}_{i,b} \approx \mathbf{M}_{i,b} \cdot \mathbf{A}_i \pmod q,$$

for all $i \in [\ell], b \in \{0,1\}$. Note that $\mathbf{A}_i \in \mathbb{Z}_q^{d\times m}$ for all $i \in [\ell]$. The symbol $\approx$ means the difference of two matrices are almost zero. Moreover, we sometimes denote $\mathbf{C}_{i,b}$ by $\mathsf{ENC}_i(\mathbf{M}_{i,b})$.

The zerotesting procedure of encoding $\mathbf{C}$ for some matrix $\mathbf{M}$ relative to the path $0 \to \ell$ is to compute

$$\mathbf{A}_0 \cdot \mathbf{C} \approx \mathbf{M} \cdot \mathbf{A}_\ell \pmod q.$$

using the published matrix $\mathbf{A}_0$, and if $\|\mathbf{A}_0 \cdot \mathbf{C}\|$ is small, determines $\mathbf{M} = \mathbf{0}_d$. Otherwise, $\mathbf{M}$ is not the zero matrix.

**Read-once branching program and (simplified) HHSS obfuscation.** A read-once branching program with a dimension $d$ for an $\ell$-bit input is the set of integral matrix $\mathcal{P} = \{\mathbf{M}_{i,b}\}_{1\le i\le \ell,\ b\in\{0,1\}} \subset \mathbb{Z}^{d\times d}$. With input $x \in \{0,1\}^\ell$, $\mathcal{P}$ can be evaluated as follows

$$\mathcal{P}(x) = \begin{cases} 0 & \text{if } \prod_{i=1}^{\ell} \mathbf{M}_{i,x_i} = \mathbf{0}_d \\ 1 & \text{otherwise,} \end{cases}$$

where $x_i$ is the $i$-th bit of $x$. The (simplified) HHSS obfuscation then consists of the set of an encoded matrix and $\mathbf{A}_0$, which is the public matrix of the GGH15 graded encoding scheme.

$$\mathcal{O}(\mathcal{P}) = \{\mathsf{ENC}_i(\mathbf{R}_{i-1}^{-1} \cdot \mathbf{M}_{i,b} \cdot \mathbf{R}_i)\} \text{ for } 1 \le i \le \ell,\ b \in \{0,1\},$$

where $\mathbf{R}_i$'s are random invertible matrices except $\mathbf{R}_0 = \mathbf{R}_\ell = \mathbf{I}_d$. More precisely, the encoding of HHSS is a $m \times m$ matrix and it is of the form of [6] graded encoding scheme

$$\mathbf{A}_{i-1} \cdot \mathsf{ENC}_i(\mathbf{R}_{i-1}^{-1} \cdot \mathbf{M}_{i,b} \cdot \mathbf{R}_i) \approx \mathbf{R}_{i-1}^{-1} \cdot \mathbf{M}_{i,b} \cdot \mathbf{R}_i \cdot \mathbf{A}_i \pmod q,$$

where the $\mathbf{A}_i$ is $d \times m$ random matrices ($m \gg d$), respectively, and $q$ is an integer. In addition, the evaluation of the encoded matrices $\mathbf{C} = \prod_{i=1}^{\ell} \mathsf{ENC}_i(\mathbf{R}_{i-1}^{-1} \cdot \mathbf{M}_{i,b} \cdot \mathbf{R}_i)$ on input $x \in \{0,1\}^\ell$ is to compute $\mathbf{A}_0 \cdot \mathbf{C}$ and it is approximately computed as follows

$$\mathbf{A}_0 \cdot \mathbf{C} \approx \prod_{i=1}^{\ell} \mathbf{M}_{i,x_i} \cdot \mathbf{A}_\ell.$$

**Left kernel attack.** To explain our idea, we describe our attack for simplified HHSS obfuscation. The detailed decryption of the left kernel attack is in Section IV.

Let $\mathcal{P} = \{\mathbf{M}_{i,b}\}$ and $\mathcal{P}' = \{\mathbf{N}_{i,b}\}$ $(1 \le i \le \ell, b \in \{0,1\})$ be two functionally equivalent read-once branching program matrices, and let $\mathcal{O}(\mathcal{X}) = \{\mathsf{ENC}_i(\mathbf{R}_{i-1}^{-1} \cdot \mathbf{X}_{i,b} \cdot \mathbf{R}_i)\}_{1\le i\le \ell, b\in\{0,1\}}$ be given encoded matrices for $\mathcal{P}$ or $\mathcal{P}'$.

Our goal is to determine whether the obfuscated program $\mathcal{OX}$ is originated by $\mathcal{P}$ or $\mathcal{P}'$. For the simplicity, we assume $\mathbf{M}_{1,1}$ and $\mathbf{N}_{1,1}$ have different left kernel. Let $\mathbf{v}$ be a short vector in the left kernel of $\mathbf{M}_{1,1}$, not in $\mathbf{N}_{1,1}$. Then, for an input $x$ satisfying $\mathcal{P}(x) = 1$, we observe the following property.

$$\mathbf{v} \cdot \mathbf{A}_0 \cdot \prod_{i=1}^{\ell} \mathsf{ENC}_i(\mathbf{R}_{i-1}^{-1} \cdot \mathbf{X}_{i,x_i} \cdot \mathbf{R}_i) \approx \mathbf{v} \cdot \prod_{i=1}^{\ell} \mathbf{X}_{i,x_i} \cdot \mathbf{A}_\ell.$$

Thus, we can distinguish between the obfuscation of two functionally equivalent read-once branching programs by calculating the norm of $\mathbf{v} \cdot \mathbf{A}_0 \cdot \prod_{i=1}^{\ell} \mathsf{ENC}_i(\mathbf{R}_{i-1}^{-1} \cdot \mathbf{X}_{i,x_i} \cdot \mathbf{R}_i)$. If the size is small, $\mathcal{X}$ equals to the program $\mathcal{P}$. Otherwise, $\mathcal{X}$ is the program $\mathcal{P}'$.

**Organization.** The preliminaries related to obfuscation are presented in Section II. The scheme description of HHSS obfuscation is discussed in Section III. Next, our distinguishing attack on HHSS obfuscation is in Section IV and finally Section V gives the conclusion.

## II. Preliminaries

**Notation.** For an integer $q \ge 2$, $\mathbb{Z}_q$ is the set of integer modulo $q$ and all integers in $\mathbb{Z}_q$ are regarded as integers in $(-q/2, q/2]$. We use the notation $\mathbb{Z}^{n\times m}$ or $\mathbb{Z}_q^{n\times m}$ to denote the set of $n \times m$ matrices over integers or $\mathbb{Z}_q$, respectively. The set $\{1, 2, \cdots, t\}$ is denoted by $[t]$ for a positive integer $t$.

Throughout this paper, we regard bold font as a vector or a matrix. Specially, $\mathbf{I}_d$ is the identity matrix of dimension $d$ and $\mathbf{0}_d$ is the $d$-dimensional zero matrix. Moreover, we sometimes abuse the notation $\mathbf{0}$ as the zero vector. The transpose of a matrix $\mathbf{A}$ is denoted by $\mathbf{A}^T$. The 2-norm of a vector $\mathbf{x}$ is denoted by $\|\mathbf{x}\|_2$ and the operator norm of a matrix $\mathbf{A} \in \mathbb{R}^{n\times m}$ is defined as $\|\mathbf{A}\|_{op} = \sup\{\|\mathbf{A}\cdot\mathbf{x}\| : \mathbf{x} \in \mathbb{R}^m \text{ with } \|\mathbf{x}\|_2 = 1\}$. It induces that the largest singular value of a matrix $\mathbf{A}$ is the same as $\|\mathbf{A}\|_{op}$ and $\|\mathbf{v}\|_{op} \le \|\mathbf{v}\|_2$ for any vector $\mathbf{v}$. For an $n$-dimensional vector $\mathbf{x}$ and an $n'$-dimensional vector $\mathbf{y}$, let $(\mathbf{x}||\mathbf{y})$ denote an $n + n'$-dimensional vector formed by concatenating $\mathbf{x}$ and $\mathbf{y}$.

**Definition 1** (Indistinguishability Obfuscation)**.** *For a security parameter $\lambda \in \mathbb{N}$, an indistinguishability obfuscation $\mathcal{O}$ is a uniform PPT machine for a circuit class $\{\mathcal{C}_\lambda\}$ which satisfies the followings:*

- *For all circuits $C \in \{\mathcal{C}_\lambda\}$ and for all input $x$,*

$$\Pr[C(x) = C'(x) : C' \leftarrow \mathcal{O}(\mathcal{C}, \lambda)] = 1$$

- *For two circuits $C_1, C_2 \in \{\mathcal{C}_\lambda\}$ having the same functionality and for any PPT distinguisher $\mathcal{D}$, there exists a negligible function* $\mathsf{negl}$ *such that*

$$|\Pr[\mathcal{D}(\mathcal{O}(C_1, \lambda) = 1] - \Pr[\mathcal{D}(\mathcal{O}(C_2, \lambda) = 1]| \le \mathsf{negl}(\lambda)$$

**Security model of *iO*.** Let $\mathcal{P}$ and $\mathcal{P}'$ be given two functionally equivalent matrix branching programs whose outputs are the same for all inputs $x$. Then, the security model of *iO* ensures that any PPT distinguisher $\mathcal{D}$ cannot distinguish whether an obfuscated program comes from $\mathcal{P}$ or $\mathcal{P}'$.

Moreover, we say "$\mathcal{O}$ does not have indistinguishability" if there exists two functionally equivalent branching programs $\mathcal{P}$, $\mathcal{P}'$ such that a PPT distinguisher $\mathcal{D}$ can determine the given obfuscated program $\mathcal{O}$ is an obfuscation of $\mathcal{P}$ or an obfuscation of $\mathcal{P}'$.

## III. HHSS OBFUSCATION

Since the advent of [3], the construction of candidate *iO* is largely composed of two steps; matrix randomization and encoding using a graded encoding scheme. To instantiate obfuscation and other various applications, three graded encoding schemes are suggested [4]–[6].

Halevi *et. al.* proposed and implemented an obfuscation [14]. They applied several randomization steps on the matrix branching program and encoded the randomized matrices by the GGH15 graded encoding scheme.

In this section, we briefly review the GGH15 graded encoding scheme and HHSS obfuscation construction. For more details, refer to [14].

### *A. GGH15 graded encoding scheme*

the GGH15 graded encoding scheme, which is a graph-induced cryptographic graded encoding scheme, is used to construct HHSS obfuscation. A directed acyclic graph $G = (V, E)$ for vertex set $V$ and edge set $E$ are used to initiate the GGH15 graded encoding scheme. In particular, we only consider a directed graph $G$ which consists of two chains of length $\ell$ with a common *source* vertex 0 and a common *sink* vertex $\ell$. They set the vertices as $V = \{0, 1, 2, \cdots, (\ell-1), 1', 2', \cdots, (\ell-1)', \ell\}$ and the edges consist of two chains defined as

$$0 \to 1 \to \cdots (\ell-1) \to \ell \quad \text{and} \quad 0 \to 1' \to \cdots (\ell-1)' \to \ell$$

$E = \{(i, (i+1)), (i', (i+1)') | i \in [\ell-2]\} \cup \{(0,1), (0,1'), ((\ell-1), \ell), ((\ell-1)', \ell)\}$.

Let $m, n$, and $q$ be integers ($n \ll m \ll q$). For each vertex $v \in V$, assign a random matrix $\mathbf{A}_v \in \mathbb{Z}_q^{n\times m}$ with its trapdoor $\tau_v$ which is needed to efficiently encode a matrix. An encoding of small plaintext $\mathbf{M} \in \mathbb{Z}^{n\times n}$ with respect to edge $u \to v$ is a small matrix $\mathbf{C} \in \mathbb{Z}_q^{m\times m}$ such that

$$\mathbf{A}_u \cdot \mathbf{C} = \mathbf{M} \cdot \mathbf{A}_v + \mathbf{E} \pmod q,$$

for a small error matrix $\mathbf{E} \in \mathbb{Z}^{n\times m}$. Note that we can easily compute a small matrix $\mathbf{C}$ using the trapdoor $\tau_u$ [23].

In HHSS obfuscation, the public parameters of the GGH15 graded encoding scheme are the graph $G = (V, E)$ and $m, n, q$ and only the source-node matrix $\mathbf{A}_0$. From the encodings and public parameters, we can compute the following with public parameters:

1) Addition: $\mathbf{C}_1 + \mathbf{C}_2$ and negation $-\mathbf{C}_1$ of two encodings $\mathbf{C}_1, \mathbf{C}_2$ with respect to edge $v \to w$
2) Multiplication: $\mathbf{C}_1 \cdot \mathbf{C}_2$ of encodings $\mathbf{C}_1, \mathbf{C}_2$ with respect to $u \to v, v \to w$, respectively.
3) Zerotesting: $\mathbf{A}_0 \cdot \mathbf{C}$ for an encoding $\mathbf{C}$ with respect to edge $0 \to \ell$

The above procedures work well in modulus $q$. In other words, the arithmetic operations addition and multiplication are homomorphic operation, and the zerotesting procedure checks whether the encoding is encoding of zero matrix or not. More specifically,

- Let $\mathbf{C}_1$ and $\mathbf{C}_2$ be two encodings of plaintext matrix $\mathbf{M}_1$ and $\mathbf{M}_2$ with respect to edge $u \to v$ respectively. *i.e.*, $\mathbf{A}_u \cdot \mathbf{C}_i = \mathbf{M}_i \cdot \mathbf{A}_v + \mathbf{E}_i \pmod q$, where norm of matrices $\mathbf{C}_i, \mathbf{E}_i, \mathbf{M}_i$ are small ($i = 1, 2$). Then, $-\mathbf{C}_1$ and $\mathbf{C}_1 + \mathbf{C}_2$ are encodings of $-\mathbf{M}_1$ and $\mathbf{M}_1 + \mathbf{M}_2$ relative to edge $u \to v$. Indeed, we observe

$$\mathbf{A}_u \cdot (-\mathbf{C}_1) = (-\mathbf{M}_1) \cdot \mathbf{A}_v - \mathbf{E}_1 \pmod q, \quad \text{and}$$
$$\mathbf{A}_u \cdot (\mathbf{C}_1 + \mathbf{C}_2) = (\mathbf{M}_1 + \mathbf{M}_2) \cdot \mathbf{A}_v + (\mathbf{E}_1 + \mathbf{E}_2) \pmod q.$$

- Let $\mathbf{C}$ and $\mathbf{C}'$ be two encodings of $\mathbf{M}$ and $\mathbf{M}'$ with respect to edge $u \to v$ and $v \to w$ respectively. In other words, $\mathbf{A}_u \cdot \mathbf{C} = \mathbf{M} \cdot \mathbf{A}_v + \mathbf{E} \pmod q$ and $\mathbf{A}_v \cdot \mathbf{C}' = \mathbf{M}' \cdot \mathbf{A}_w + \mathbf{E}' \pmod q$ with the small matrices $\mathbf{C}, \mathbf{C}', \mathbf{E}, \mathbf{E}', \mathbf{M}$, and $\mathbf{M}'$. Then we have

$$\begin{aligned} \mathbf{A}_u \cdot (\mathbf{C} \cdot \mathbf{C}') &= (\mathbf{M} \cdot \mathbf{A}_v + \mathbf{E}) \cdot \mathbf{C}' \\ &= \mathbf{M} \cdot \mathbf{A}_v \cdot \mathbf{C}' + \mathbf{E} \cdot \mathbf{C}' \pmod q \\ &= \mathbf{M} \cdot (\mathbf{M}' \cdot \mathbf{A}_w + \mathbf{E}') + \mathbf{E} \cdot \mathbf{C}' \pmod q \\ &= (\mathbf{M} \cdot \mathbf{M}') \cdot \mathbf{A}_w + \mathbf{M} \cdot \mathbf{E}' + \mathbf{E} \cdot \mathbf{C}' \pmod q \end{aligned}$$

  Note that $\mathbf{C}_1 + \mathbf{C}_2$, $\mathbf{M}_1 + \mathbf{M}_2$, and $\mathbf{E}_1 + \mathbf{E}_2$ are small. Moreover, $\mathbf{M} \cdot \mathbf{M}'$, $\mathbf{C} \cdot \mathbf{C}'$, and $\mathbf{M} \cdot \mathbf{E}' + \mathbf{E} \cdot \mathbf{C}'$ are still small.
- The zerotesting procedure can determine whether a plaintext matrix $\mathbf{M}$ is zero or not by estimating $\|\mathbf{A}_0 \cdot \mathbf{C}\|_{op}$ for a given encoding $\mathbf{C}$ of $\mathbf{M}$ with respect to edge $0 \to \ell$. Specially, if $\|\mathbf{A}_0 \cdot \mathbf{C}\|_{op} \le q/2^{10}$, $\mathbf{C}$ is an encoding matrix of $\mathbf{M} = \mathbf{0}_n$.

### *B. Construction of HHSS Obfuscation*

**Randomizing Branching Program.** As noted above, HHSS obfuscation only support the read-once branching programs since they remove *scalar bundling* [2].

Halevi *et. al.* use two randomization steps called higher-dimensional embedding and Kilian-style randomization upon a given read-once branching program $\mathcal{P}$. For $i \in [\ell]$ and $b \in \{0, 1\}$, they first embed the matrix $\mathbf{M}_{i,b}$ into a $n$-dimensional matrix which is a block diagonal matrix of the form $\mathsf{diag}(\mathbf{M}_{i,b}, \mathbf{R}_{i,b})$, where $\mathbf{R}_{i,b}$ is a $d' \times d'$ random (small) matrix. In [14], they picked $d' = \lceil \sqrt{\lambda/2} \rceil$. After higher-dimensional embedding, they apply Kilian-style randomization

[2]Scalar bundling is used to capture the *mixed-input attack*, which uses invalid inputs of the matrix branching program. However, read-once programs are not threatened by this attack.

to the matrix $\mathsf{diag}(\mathbf{M}_{i,b}, \mathbf{R}_{i,b})$ to make it look like a random matrix. In other words, the randomized matrix $\tilde{\mathbf{M}}_{i,b}$ is of the form

$$\tilde{\mathbf{M}}_{i,b} = \mathbf{S}_{i-1}^{-1} \cdot \mathsf{diag}(\mathbf{M}_{i,b}, \mathbf{R}_{i,b}) \cdot \mathbf{S}_i, \quad i \in [\ell], b \in \{0,1\},$$

where $\mathbf{S}_i \in \mathbb{Z}^{n\times n}$ is a random invertible matrix and $\mathbf{S}_0$ and $\mathbf{S}_\ell$ are identity matrices over dimension $n$.

Halevi *et. al.* employ an additional structure which is called the *dummy program* for the zerotesting procedure. It consists of $\ell$ pairs of $d \times d$ binary matrices $\mathbf{M'}_{i,b}$, where $i \in [\ell]$ and $b \in \{0,1\}$. More precisely, the first matrix $\mathbf{M'}_{1,b}$ and the last matrix $\mathbf{M'}_{\ell,b}$ in the dummy program are of the form $\mathsf{diag}(\mathbf{I}_{\lfloor d/2 \rfloor}, \mathbf{0}_{\lceil d/2 \rceil})$ and $\mathsf{diag}(\mathbf{0}_{\lfloor d/2 \rfloor}, \mathbf{I}_{\lceil d/2 \rceil})$ respectively. Other matrices $\mathbf{M'}_{i,b}$ are set to $\mathbf{I}_d$ so that the product $\prod_{i=1}^{\ell} \mathbf{M'}_{i,b}$ is always the zero matrix. A dummy program should be also randomized like a branching program with the same random matrix $\mathbf{R}_{i,b}$ in the lower-right quadrant. Namely, a randomized matrix $\tilde{\mathbf{M}}'_{i,b}$ of $\mathbf{M'}_{i,b}$ is of the form

$$\tilde{\mathbf{M}}'_{i,b} = \mathbf{S'}_{i-1}^{-1} \cdot \mathsf{diag}(\mathbf{M'}_{i,b}, \mathbf{R}_{i,b}) \cdot \mathbf{S'}_i, \qquad i \in [\ell], b \in \{0,1\},$$

where $\mathbf{S'}_i \in \mathbb{Z}^{n\times n}$ is a random invertible matrix and $\mathbf{S'}_0$ and $\mathbf{S'}_\ell$ are identity matrices.

Randomizing techniques are conducted to branching program and dummy program, and their functionalities are invariant. In other words, $\prod_{i=1}^{\ell} \tilde{\mathbf{M}}_{i,x_i} - \prod_{i=1}^{\ell} \tilde{\mathbf{M}}'_{i,x_i}$ is zero when $\prod_{i=1}^{\ell} \mathbf{M}_{i,x_i}$ is also zero. Otherwise, $\prod_{i=1}^{\ell} \mathbf{M}_{i,x_i}$ is not zero.

**Encoding using the the GGH15 graded encoding scheme.** We describe how to employ the GGH15 graded encoding scheme to encode randomized matrices.

After randomization steps, we recall the setting of the GGH15 graded encoding scheme. Specifically, $G = (V, E)$ is a two-chain graph and let $m, n$, and $q$ be integers ($d + d' = n \ll m \ll q$). We sample random matrices $\mathbf{A}_i \in \mathbb{Z}_q^{n\times m}$ and $\mathbf{A'}_i \in \mathbb{Z}_q^{n\times m}$ with their trapdoors corresponding vertices $i \in \{1, 2, \cdots, (\ell-1)\}$ and $i' \in \{1', 2', \cdots, (\ell-1)'\}$, respectively. Moreover, random matrices $\mathbf{A}_0$ and $\mathbf{A}_\ell$ are assigned in a source node 0 and a sink node $\ell$, respectively. In that case, we define $\mathbf{A}_0 = \mathbf{A}'_0$ and $\mathbf{A}_\ell = \mathbf{A}'_\ell$ for convenience.

For each $i \in \{0, 1, 2, \cdots, \ell\}$, we compute two matrices $\tilde{\mathbf{M}}_{i,b}$ relative to an edge $(i-1) \to i$ and $\tilde{\mathbf{M}}'_{i,b}$ relative to an edge $(i-1)' \to i'$.

*i.e.*, we have

$$\mathbf{A}_i \cdot \mathbf{C}_{i,b} = \tilde{\mathbf{M}}_{i,b} \cdot \mathbf{A}_{i+1} + \mathbf{E}_{i,b} \pmod q$$
$$\text{and } \mathbf{A'}_i \cdot \mathbf{C'}_{i,b} = \tilde{\mathbf{M}}'_{i,b} \cdot \mathbf{A'}_{i+1} + \mathbf{E'}_{i,b} \pmod q,$$

for some small errors $\mathbf{E}_{i,b} \in \mathbb{Z}^{n\times m}$ and $\mathbf{E'}_{i,b} \in \mathbb{Z}^{n\times m}$.

In addition, Halevi *et. al.* utilize an additional structure, called *outer safeguard*, to make attacks harder since its security is suspect. The outer safeguard uses a Kilian style randomization again to the output of encodings. Applying the outer safeguard step, we have

$$\hat{\mathbf{C}}_{i,b} = \mathbf{P}_{i-1}^{-1} \cdot \mathbf{C}_{i,b} \cdot \mathbf{P}_i \quad \text{and} \quad \hat{\mathbf{C}}'_{i,b} = \mathbf{P'}_{i-1}^{-1} \cdot \mathbf{C'}_{i,b} \cdot \mathbf{P'}_i$$

for some random invertible small matrices $\mathbf{P}_0, \cdots, \mathbf{P}_\ell$ and $\mathbf{P'}_0, \cdots, \mathbf{P'}_\ell$ with $\mathbf{P}_0 = \mathbf{P}_\ell = \mathbf{P'}_0 = \mathbf{P'}_\ell = \mathbf{I}_m$. Note that a new encoding technique called safeguard does not affect the output of zerotesting procedure because of telescopic cancellation and the two matrices corresponding to source and sink are identity matrices. Hence, the obfuscation consists of the following matrices:

$$\mathcal{O}(\mathcal{P}) = \left( \mathbf{A}_0, \{\hat{\mathbf{C}}_{i,b}\hat{\mathbf{C}}'_{i,b}\}_{i\in[\ell], b\in\{0,1\}} \right).$$

**Evaluation of Obfuscation.** For an input $x \in \{0,1\}^\ell$, the first step of evaluation is to compute $\mathbf{A}_0 \cdot \left( \prod_{i=1}^{\ell} \hat{\mathbf{C}}_{i,x_i} - \prod_{i=1}^{\ell} \hat{\mathbf{C}}'_{i,x_i} \right)$. Since it is of the form

$$\begin{pmatrix} \prod_{i=1}^{\ell} \mathbf{M}_{i,x_i} & \\ & \mathbf{0}_{d'} \end{pmatrix} \cdot \mathbf{A}_\ell + \mathsf{Error},$$

we can determine whether or not a matrix $\prod_{i=1}^{\ell} \mathbf{M}_{i,x_i}$ is the zero matrix from its norm. In other words, when $\prod_{i=1}^{\ell} \mathbf{M}_{i,x_i}$ is zero $\mathbf{A}_0 \cdot \left( \prod_{i=1}^{\ell} \hat{\mathbf{C}}_{i,x_i} - \prod_{i=1}^{\ell} \hat{\mathbf{C}}'_{i,x_i} \right)$ is of the form $\mathsf{Error} - \mathsf{Error}'$ and thus sufficiently small.

We abuse some notations to describe precisely. For each vertex $i$, $\mathbf{C}_i$ denotes an encoding of plaintext $\tilde{\mathbf{M}}_i = \mathbf{S}_{i-1}^{-1} \cdot \mathbf{M}_i \cdot \mathbf{S}_i$ relative to a path $(i-1) \to i$, and $\mathbf{C} = \prod_{i=1}^{\ell} \mathbf{C}_i$ denotes an encoding relative to path $0 \to \ell$. Then, we have $\mathbf{A}_0 \cdot \mathbf{C} = \left( \prod_{i=1}^{\ell} \tilde{\mathbf{M}}_i \right) \cdot \mathbf{A}_\ell + \mathsf{Error} \pmod q$, and $\mathsf{Error}$ is of the form

$$\sum_{j=1}^{\ell} \left( \prod_{i=1}^{j-1} \mathbf{M}_i \right) \cdot \mathbf{S}_{j-1} \cdot \mathbf{E}_j \cdot \left( \prod_{i=j+1}^{\ell} \mathbf{C}_i \right).$$

The size of the $\mathsf{Error}$ term largely depends on the term $\mathbf{E}_1 \cdot \prod_{i=2}^{\ell} \mathbf{C}_i$ since the size of matrix $\mathbf{C}_i$ is larger than that of other $\tilde{\mathbf{M}}_i$, $\mathbf{S}_{j-1}$ and noise term $\mathbf{E}_j$ for all $i, j$. Therefore, we can speculate the norm $\mathsf{Error}$ from the construction of trapdoor sampling and error size $\mathbf{E}_i$,

$$\left\| \mathbf{E}_1 \cdot \prod_{j=2}^{\ell} \mathbf{C}_j \right\|_{op} \approx 2^7 \cdot \sigma_x^{\ell-1} \cdot m^{\ell/2} \cdot 2^{\ell-1},$$

where $\sigma_x$ is a parameter of spherical Gaussian distribution used in the trapdoor sampling. For more details of parameters for trapdoor sampling and error size, refer to Section 5 in [14] or Appendix.

Halevi *et.al.* designed a zerotesting procedure by estimating $\|\mathbf{A}_0 \cdot \mathbf{C}\|$. Namely, the obfuscated program $\mathcal{O}(\mathcal{P})(x)$ outputs 0 when $\|\mathbf{A}_0 \cdot \mathbf{C}\| \le q/2^{10}$. Similarly, $\mathcal{O}(\mathcal{P})(x)$ outputs 1 when $\|\mathbf{A}_0 \cdot \mathbf{C}\| > q/2^{10}$. Hence, for the correctness of the obfuscation program, the following equation should hold.

$$\log q \geq 7 + \log \sigma_x \cdot (\ell - 1) + \log m \cdot \ell/2 + (\ell - 1) + 10.$$

In summary, we obtain a circuit $\mathcal{O}(BP)$ such that

$$\mathcal{O}(\mathcal{P})(x)^3 = \begin{cases} 0 & \text{if } \|\mathbf{A}_0 \cdot \mathbf{C}\|_{op} \leq q/2^{10}, \\ 1 & \text{othersiwe.} \end{cases}$$

**Remark.** To reduce the size of $\mathcal{O}(\mathcal{P})$, Halevi *et. al.* proposed a special encoding for sink $\ell$. Specially, $\mathbf{A}_\ell$ is a vector in $\mathbb{Z}_q^{n \times 1}$ and $\mathbf{E}_\ell$ is also a small vector in $\mathbb{Z}_q^{m \times 1}$. If someone wants to encode a small plaintext matrix $\mathbf{M}$ with respect to a path $\ell - 1 \to \ell$, compute $[\mathbf{M} \cdot \mathbf{A}_\ell + \mathbf{E}_\ell]_q$, and sample a small vector $\mathbf{C}$ such that $\mathbf{A}_{\ell-1} \cdot \mathbf{C} = \mathbf{M} \cdot \mathbf{A}_\ell + \mathbf{E}_\ell$ using a trapdoor to sample a small vector, and finally output $\hat{\mathbf{C}} = \mathbf{P}_{\ell-1} \cdot \mathbf{C}_\ell$.

## IV. Cryptanalysis of the HHSS Obfuscation

In this section we present our attack, which is called *left kernel attack*, and further analysis of HHSS obfuscation.

Two functionally equivalent branching programs $\{\mathbf{M}_{i,b}, \mathbf{N}_{i,b}\}_{i \in [\ell], b \in \{0,1\}}$ are given. We found that the left-kernel of plaintext matrices leads distinguishing attack to obfuscated program in the zerotesting procedure. More precisely, we observe the upper quadrant of the matrix $\mathbf{A}_0 \cdot \mathbf{C}$ is the product of plaintext matrices related to an encoding matrix $\mathbf{C}$. Hence, by using a left kernel vector of the branching program matrix, we can disclose some hidden information from the obfuscated program.

### *A. Description of Our Attack.*

Let $\mathcal{P}$ and $\mathcal{P}'$ be two functionally equivalent read-once matrix branching programs for $\ell$-bit input corresponding $\mathbf{M}_{i,b} \in \mathbb{Z}^{d \times d}$ and $\mathbf{N}_{i,b} \in \mathbb{Z}^{d \times d}$ for $i \in [\ell], b \in \{0,1\}$ respectively. Also, we have a program $\mathcal{O}$ encoded by the GGH15 graded encoding scheme, but we do not know whether or not $\mathcal{O}$ is an obfuscation of the branching program $\mathcal{P}$ or $\mathcal{P}'$. We want to determine whether an obfuscated program $\mathcal{O}$ comes from $\mathcal{P}$ or not.

Suppose we have public matrix $\mathbf{A}_0$ of the GGH15 graded encoding scheme and following matrices.

$$\mathcal{P} = \{\mathbf{M}_{i,b}\}_{i \in [\ell], b \in \{0,1\}}, \ \mathcal{P}' = \{\mathbf{N}_{i,b}\}_{i \in [\ell], b \in \{0,1\}}, \\ \mathcal{O} = \{\hat{\mathbf{C}}_{i,b}, \hat{\mathbf{C}}'_{i,b}\}_{i \in [\ell], b \in \{0,1\}},$$

where $\hat{\mathbf{C}}_{i,b}$ and $\hat{\mathbf{C}}'_{i,b}$ are encodings of a matrix and a dummy branching program relative to paths $(i-1) \to i$ and $(i-1)' \to i'$ using the GGH15 graded encoding scheme, respectively. Note that matrices $\hat{\mathbf{C}}_{i,b}, \hat{\mathbf{C}}'_{i,b}$ are $m \times m$ integer matrices with $m \geq d$.

For some input $x$ such that $\mathcal{P}(x) = 0$ and $\prod_{i=1}^{\ell} \mathbf{M}_{i,x_i} = \mathbf{0}_d$, there exists at least one singular matrix $\mathbf{M}_{i,x_i}$ for some $i$. Let $\mathbf{M}_{1,1}$ be a singular matrix for convenience.[4] Thus, there is a nonzero vector $\mathbf{v} \in \mathbb{Z}^d$ such that $\mathbf{v} \cdot \mathbf{M}_{1,1} = \mathbf{0}_d$. Assume that we can find a short vector $\mathbf{v}$ such that $\mathbf{v} \cdot \mathbf{M}_{1,1} = \mathbf{0}_d$ and $\mathbf{v} \cdot \mathbf{N}_{1,1} \neq \mathbf{0}_d$. Then, we are able to distinguish an obfuscated program by employing a vector $\mathbf{v} \in \mathbb{Z}^d$.

From now, let an input $x$ be represented as $x_1 x_2 \cdots x_n$ with $x_1 = 1$ such that $\mathcal{P}(x) = 1$ and $\mathcal{P}'(x) = 1$. Then, we observe

$$\mathbf{v} \cdot \prod_{i=1}^{\ell} \mathbf{M}_{i,x_i} = \mathbf{0}_d \ \text{ and } \ \mathbf{v} \cdot \prod_{i=1}^{\ell} \mathbf{N}_{i,x_i} \neq \mathbf{0}_d \text{ with high probability.}$$

Let $\tilde{\mathbf{M}}_{i,b}$ and $\tilde{\mathbf{N}}_{i,b}$ (with $i \in [\ell], b \in \{0,1\}$) be randomized matrices of $\mathbf{M}_{i,b}$ and $\mathbf{N}_{i,b}$, respectively. If $\hat{\mathbf{C}}$ is an encoding of $\prod_{i=1}^{\ell} \tilde{\mathbf{M}}_{i,x_i}$ relative to the path $0 \to \ell$, then we have

$$\begin{aligned} \hat{\mathbf{v}} \cdot (\mathbf{A}_0 \cdot \hat{\mathbf{C}}) &= \hat{\mathbf{v}} \cdot \left( \prod_{i=1}^{\ell} \tilde{\mathbf{M}}_{i,x_i} \cdot \mathbf{A}_\ell + \mathsf{Error} \right) \pmod q \\ &= \left( \hat{\mathbf{v}} \cdot \prod_{i=1}^{\ell} \tilde{\mathbf{M}}_{i,x_i} \right) \cdot \mathbf{A}_\ell + \hat{\mathbf{v}} \cdot \mathsf{Error} \pmod q \\ &= \hat{\mathbf{v}} \cdot \mathsf{Error} \pmod q, \end{aligned}$$

where $\hat{\mathbf{v}} = (\mathbf{v}||\mathbf{0})$ is an $n \ (= d + d')$-dimensional vector. Note that $\|\hat{\mathbf{v}} \cdot \mathsf{Error} \pmod q\|$ must be small if $\|\hat{\mathbf{v}}\|$ is small. Moreover, if $\hat{\mathbf{C}}$ comes from a plaintext matrix $\prod_{i=1}^{\ell} \tilde{\mathbf{N}}_{i,x_i}$, then we have

$$\begin{aligned} \hat{\mathbf{v}} \cdot (\mathbf{A}_0 \cdot \hat{\mathbf{C}}) &= \hat{\mathbf{v}} \cdot \left( \prod_{i=1}^{\ell} \tilde{\mathbf{N}}_{i,x_i} \cdot \mathbf{A}_\ell + \mathsf{Error} \right) \pmod q \\ &= \left( \hat{\mathbf{v}} \cdot \prod_{i=1}^{\ell} \tilde{\mathbf{N}}_{i,x_i} \right) \cdot \mathbf{A}_\ell + \hat{\mathbf{v}} \cdot \mathsf{Error} \pmod q. \end{aligned}$$

Note that $\|\hat{\mathbf{v}} \cdot (\mathbf{A}_0 \cdot \hat{\mathbf{C}})\|$ looks like a random over $\mathbb{Z}_q$ since $(\hat{\mathbf{v}} \cdot \prod_{i=1}^{\ell} \tilde{\mathbf{N}}_{i,x_i}) \cdot \mathbf{A}_\ell$ does not vanish. Therefore, a distinguisher $\mathcal{D}$ can output

$$\mathcal{D}(\mathcal{O}) = \begin{cases} \mathcal{P} & \text{if } \|\hat{\mathbf{v}} \cdot (\mathbf{A}_0 \cdot \hat{\mathbf{C}})\|_{op} \leq q/2^{10}. \\ \mathcal{P}' & \text{otherwise.} \end{cases}$$

**Remark.** Some programs do not exist a short vector $\mathbf{v}$ such that $\hat{\mathbf{v}} \cdot \mathbf{M}_{1,x_1} = \mathbf{0}_d$, for example, a full rank matrix $\mathbf{M}_{1,x_1}$ does not have a nonzero left kernel vector. However, in that case we can still apply our attacks by employing a matrix $\mathbf{M}^{(1)} = \begin{pmatrix} \mathbf{M}_{1,1} \\ \mathbf{M}_{1,0} \end{pmatrix}$ instead of $\mathbf{M}_{1,1}$. $\mathbf{M}^{(1)}$ must have a nontrivial left kernel because its rank is $d$ with high probability.

### *B. Examples of Attackable Branching Programs*

In this section, we present two concrete examples to understand our attack.

**Toy Example.** We here give two simple equivalent branching programs and the results of our attack with specific setting of implementation. Two branching programs $\mathcal{P}, \mathcal{P}'$, which satisfy $\mathcal{P}(x) = \mathcal{P}'(x) = 0$ for $x = 01$ and $\mathcal{P}(x) = \mathcal{P}'(x) = 1$ otherwise, are given as follows

[3]In their implementation, however, the obfuscated program outputs $1 - \mathcal{O}(\mathcal{P})(x)$.

[4]Our attack also work well for the other cases. See remark.

| |
|---|
| $$\mathbf{M}_{1,0}=\begin{pmatrix}1&0\\1&0\end{pmatrix},\quad \mathbf{M}_{2,0}=\begin{pmatrix}1&0\\0&0\end{pmatrix},$$ $$\mathbf{M}_{1,1}=\begin{pmatrix}0&0\\1&1\end{pmatrix},\quad \mathbf{M}_{2,1}=\begin{pmatrix}0&0\\0&1\end{pmatrix}.$$ Matrix branching program $\mathcal{P}$ |
| $$\mathbf{N}_{1,0}=\begin{pmatrix}1&0\\1&0\end{pmatrix},\quad \mathbf{N}_{2,0}=\begin{pmatrix}1&0\\1&1\end{pmatrix},$$ $$\mathbf{N}_{1,1}=\begin{pmatrix}0&1\\0&1\end{pmatrix},\quad \mathbf{N}_{2,1}=\begin{pmatrix}0&0\\1&1\end{pmatrix}.$$ Matrix branching program $\mathcal{P}'$ |

TABLE I: Toy example

Note that for two left kernel vectors $\mathbf{v}=(1,0)$ and $\mathbf{w}=(1,-1)$, the branching programs satisfy

$$\mathbf{v}\cdot\mathbf{M}_{1,1}=\mathbf{0},\qquad \mathbf{v}\cdot\mathbf{N}_{1,1}\neq\mathbf{0}$$
$$\mathbf{w}\cdot\mathbf{N}_{1,1}=\mathbf{0},\qquad \mathbf{w}\cdot\mathbf{M}_{1,1}\neq\mathbf{0},$$

respectively. If we set the security parameter $\lambda=80$ then the embedding dimension $d'=5^5$ and $q=34009074817199$ and $\epsilon=10$. Let $\{\hat{\mathbf{C}}_{i,b}\}$ and $\{\hat{\mathbf{D}}_{i,b}\}$ be the obfuscated programs of $\mathbf{M}$ and $\mathbf{N}$, respectively. The evaluation vectors are computed as follows:

$$\mathbf{A}_0\cdot\hat{\mathbf{C}}_{1,1}\cdot\hat{\mathbf{C}}_{2,0}=\begin{pmatrix}-10884489\\6341880489273\\-2809809111564\\-9752397591639\\-3576418758634\\-7724278907092\\11995182501421\end{pmatrix},$$

$$\mathbf{A}_0\cdot\hat{\mathbf{D}}_{1,1}\cdot\hat{\mathbf{D}}_{2,0}=\begin{pmatrix}-281520684456\\-281549139175\\-410308959247\\-15011968960796\\-5708322256588\\-2185335626767\\-593005703340\end{pmatrix}.$$

At last, the inner product values of the extended left kernel vectors $\hat{\mathbf{v}}=(1,0||\mathbf{0}),\hat{\mathbf{w}}=(1,-1||\mathbf{0})$ and the evaluation vectors are

$$\begin{aligned}z_{\mathbf{v},\mathbf{C}}&=\hat{\mathbf{v}}\cdot\mathbf{A}_0\cdot\hat{\mathbf{C}}_{1,1}\cdot\hat{\mathbf{C}}_{2,0}=-10884489\\z_{\mathbf{v},\mathbf{D}}&=\hat{\mathbf{v}}\cdot\mathbf{A}_0\cdot\hat{\mathbf{D}}_{1,1}\cdot\hat{\mathbf{D}}_{2,0}=-281520684456\\z_{\mathbf{w},\mathbf{C}}&=\hat{\mathbf{w}}\cdot\mathbf{A}_0\cdot\hat{\mathbf{C}}_{1,1}\cdot\hat{\mathbf{C}}_{2,0}=-6341891373762\\z_{\mathbf{w},\mathbf{D}}&=\hat{\mathbf{w}}\cdot\mathbf{A}_0\cdot\hat{\mathbf{D}}_{1,1}\cdot\hat{\mathbf{D}}_{2,0}=28454719.\end{aligned}$$

We can easily observe that the inner products are fairly small if the left kernel vector corresponds to the obfuscated program.

[5]In the original paper, authors recommend that it should be $d'=7$ to satisfy $d'=\lceil\sqrt{\lambda/2}\rceil$, but their implementation chooses $d'=5$.

Actually the values $z_{\mathbf{v},\mathbf{C}},z_{\mathbf{w},\mathbf{D}}$ are less than $2^{-20}\cdot q$, whereas $z_{\mathbf{v},\mathbf{D}}$ and $z_{\mathbf{w},\mathbf{C}}$ are larger than $2^{-10}\cdot q$. Therefore, the left kernel attack works well for this example.

**Point Functions.** Now we show more complex and lengthy examples of functionally equivalent branching programs. We denote an $n\times n$ diagonal matrix with $i$-th diagonal element $a_i$ by $\mathsf{diag}(\mathbf{a})$ for vector $\vec{a}=(a_1,\cdots,a_n)$. We also denote an $n$ dimensional all-1 vector by $\mathbf{1}_n$, and $n$ dimensional standard basis by $\{\mathbf{e}_i\}_{1\le i\le n}$. Then two branching programs $\mathcal{P},\mathcal{P}'$ placed in Table 1 satisfies $\mathcal{P}(0)=\mathcal{P}'(0)=0$ and $\mathcal{P}(x)=\mathcal{P}'(x)=1$ for all $x\in\mathbb{Z}_2^n\setminus\{0\}$.

| |
|---|
| $\mathbf{M}_{1,0}=\mathsf{diag}(\mathbf{1}_{n+1}-\mathbf{e}_1-\mathbf{e}_{n+1})$, $\mathbf{M}_{i,0}=\mathsf{diag}(\mathbf{1}_{n+1}-\mathbf{e}_i)$ for $2\le i\le n$, $\mathbf{M}_{i,1}=\mathsf{diag}(\mathbf{1}_{n+1})$ for $1\le i\le n$. Matrix branching program $\mathcal{P}$ |
| $\mathbf{N}_{1,0}=\mathsf{diag}(\mathbf{1}_{n+1}-\mathbf{e}_1-\mathbf{e}_2)$, $\mathbf{N}_{i,0}=\mathsf{diag}(\mathbf{1}_{n+1}-\mathbf{e}_{i+1})$ for $2\le i\le n$, $\mathbf{N}_{i,1}=\mathsf{diag}(\mathbf{1}_{n+1})$ for $1\le i\le n$. Matrix branching program $\mathcal{P}'$ |

TABLE II: Two point function branching programs

These two programs clearly have different left kernel. For example, $\mathbf{v}=\mathbf{e}_{n+1}$ is a left kernel vector of $\mathbf{M}_{1,0}$ whereas $\mathbf{v}\cdot\mathbf{N}_{1,0}=\mathbf{e}_{n+1}$. Therefore the obfuscations of $\mathcal{P},\mathcal{P}'$ would be distinguished by our attack.

### C. expansion of attack for other safeguards

As presented in Section III-B, we assume that the randomized program $\{\tilde{\mathbf{M}}_{i,b}\}$, the dummy program $\{\tilde{\mathbf{M}}'_{i,b}\}$, and a source node matrix $\mathbf{A}_0$ are given in the following form for the branching program $\{\mathbf{M}_{i,b}\}$ .

$$\tilde{\mathbf{M}}_{i,b}=\mathbf{S}_{i-1}^{-1}\cdot\mathsf{diag}(\mathbf{M}_{i,b},\mathbf{R}_{i,b})\cdot\mathbf{S}_i,\quad i\in[\ell],b\in\{0,1\},$$

$$\tilde{\mathbf{M}}'_{i,b}=\mathbf{S}'^{-1}_{i-1}\cdot\mathsf{diag}(\mathbf{M}'_{i,b},\mathbf{R}_{i,b})\cdot\mathbf{S}'_i,\quad i\in[\ell],b\in\{0,1\},$$

Except for the HHSS obfuscation, some obfuscations over graded encoding schemes carry out an additional randomization process such as scalar bundling and bookend vectors. In this section, we explain how the obfuscated program changes in each randomization process and explain how to attack obfuscations accordingly.

**Case 1: Add scalar bundling.** In the case of scalar bundling, the randomization of branching program is changed as follows.

$$\tilde{\mathbf{M}}_{i,b}=\mathbf{S}_{i-1}^{-1}\cdot\mathsf{diag}(\alpha_{i,b}\cdot\mathbf{M}_{i,b},\mathbf{R}_{i,b})\cdot\mathbf{S}_i,\quad i\in[\ell],b\in\{0,1\},$$

$$\tilde{\mathbf{M}}'_{i,b}=\mathbf{S}'^{-1}_{i-1}\cdot\mathsf{diag}(\alpha'_{i,b}\cdot\mathbf{M}'_{i,b},\mathbf{R}_{i,b})\cdot\mathbf{S}'_i,\quad i\in[\ell],b\in\{0,1\},$$

where the $\alpha_{i,b}, \alpha'_{i,b}$ ($i \in [\ell], b \in \{0,1\}$) are elements of a ring that contains an integer set and satisfying $\prod_{i=1}^{\ell} \alpha_{i,b} = \prod_{i=1}^{\ell} \alpha'_{i,b}$ for all $b \in \{0,1\}$. Obviously, this modification does not affect their functionalities. In other words, $\prod_{i=1}^{\ell} \tilde{\mathbf{M}}_{i,x_i} - \prod_{i=1}^{\ell} \tilde{\mathbf{M}}'_{i,x_i}$ is zero when $\prod_{i=1}^{\ell} \mathbf{M}_{i,x_i}$ is also zero. Otherwise, $\prod_{i=1}^{\ell} \mathbf{M}_{i,x_i}$ is not zero.

Since the matrix singularity is invariant by scalar multiplication, this modification also does not affect our attacks. More precisely, for an input $x$ such that $\mathcal{P}(x) = 0$ and $\prod_{i=1}^{\ell} \mathbf{M}_{i,x_i} = \mathbf{0}_d$, we assume that $\mathbf{M}_{1,1}$ be a singular matrix. Thus, there exists a nonzero $\mathbf{v} \in \mathbb{Z}^d$ such that $\mathbf{v} \cdot \mathbf{M}_{1,1} = \mathbf{0}$. It implies that $(\mathbf{v}||\mathbf{0}) \cdot \tilde{\mathbf{M}}_{1,1} = \mathbf{0}$. After this, we can get the same conclusion through the attack process as in section IV-A.

**Case 2: Add bookend vector.** In this case, instead of a branching program, the source node matrix $\mathbf{A}_0$ is replaced by a vector. It means that we have public vector $\mathbf{A}_0 \in \mathbb{Z}^m$ and following matrices.

$$\mathcal{P} = \{\mathbf{M}_{i,b}\}_{i\in[\ell],b\in\{0,1\}},\ \mathcal{P}' = \{\mathbf{N}_{i,b}\}_{i\in[\ell],b\in\{0,1\}},$$
$$\mathcal{O} = \{\hat{\mathbf{C}}_{i,b}, \hat{\mathbf{C}}'_{i,b}\}_{i\in[\ell],b\in\{0,1\}},$$

where $\hat{\mathbf{C}}_{i,b}$ and $\hat{\mathbf{C}}'_{i,b}$ are encodings of a branching program matrix and a dummy program relative to paths $(i-1) \to i$ and $(i-1)' \to i'$ using the GGH15 graded encoding scheme, respectively. Therefore, it is no longer valid to multiply vector $\hat{\mathbf{v}}$ by $\mathbf{A}_0 \cdot \prod_{i=1}^{\ell} \tilde{\mathbf{M}}_{i,b}$ for $\mathbf{v} \in \ker(\mathbf{M}_{1,1})$, as in Section IV-A.

Instead, we employ the attack called matrix zeroizing attack proposed by Cheon *et. al.* [18]. First, we assume $2^\ell$ is larger than $n^2$. Then, the matrix set $\{\mathbf{B}_{b_1,\cdots,b_\ell} := \prod_{i=1}^{\ell} \mathbf{M}_{i,b_i} :\ b_i \in \{0,1\}\}$ is dependent set. For the sake of simplicity, we denote the index set $b_1, \cdots, b_\ell$ as $\mathbf{b}$. Then, there exists integers $\mathbf{v_b} \in \mathbb{Z}$ such that $\sum \mathbf{v_b} \cdot \mathbf{B_b} = \mathbf{0} \in \mathbb{Z}^{d\times d}$. We additionally assume that the integers $\mathbf{v_b}$ are small and satisfy $\sum \mathbf{v_b} \cdot \prod_{i=1}^{\ell} \mathbf{N}_{i,b_i} \neq \mathbf{0}$.

Now if $\mathcal{O}$ is an obfuscation of $\mathcal{P}$, we have

$$\begin{aligned}
&\sum \mathbf{v_b} \cdot \mathbf{A}_0 \cdot \left(\prod_{i=1}^{\ell} \hat{\mathbf{C}}_{i,b_i} - \prod_{i=1}^{\ell} \hat{\mathbf{C}}'_{i,b_i}\right) \\
&= \sum \mathbf{v_b} \cdot \left(\prod_{i=1}^{\ell} \mathsf{diag}(\mathbf{M}_{i,b_i}, \mathbf{0}) \cdot \mathbf{A}_\ell + \mathsf{Error}\right) \pmod q \\
&= \mathsf{diag}(\mathbf{0},\mathbf{0}) \cdot \mathbf{A}_\ell + \sum \mathbf{v_b} \cdot \mathsf{Error} \pmod q \\
&= \sum \mathbf{v_b} \cdot \mathsf{Error} \pmod q.
\end{aligned}$$

Note that $\|\sum \mathbf{v_b} \cdot \mathsf{Error} \pmod q\|$ must be small if each $\|\mathbf{v_b}\|$ is small. From this property one can distinguish whether the given obfuscation program is $\mathcal{P}$ or not.

## V. Conclusion

In this paper, we present a polynomial time attack for HHSS obfuscation of some functionally equivalent branching programs. We can efficiently distinguish between obfuscations of branching programs from the left kernel of the branching program matrices. Since our attack only depends on the left kernel of the matrix, it is not necessary to obtain several encodings of zero from evaluations. Therefore, two evasive functions obfuscated by HHSS obfuscation are distinguishable although anyone can hardly obtain encodings of zero. We also observe that the variants of HHSS obfuscation of evasive functions with the portion of safeguards are distinguishable. However, distinguishing between the GGH15 obfuscations of evasive functions with all the safeguards remains as an open problem.

## References

[1] S. Goldwasser and G. N. Rothblum, "On best-possible obfuscation," in *Theory of Cryptography, 4th Theory of Cryptography Conference, TCC 2007, Amsterdam, The Netherlands, February 21-24, 2007, Proceedings*, 2007, pp. 194–213. [Online]. Available: https://doi.org/10.1007/978-3-540-70936-7_11

[2] B. Barak, "Hopes, fears, and software obfuscation," *Commun. ACM*, vol. 59, no. 3, pp. 88–96, 2016. [Online]. Available: http://doi.acm.org/10.1145/2757276

[3] S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, and B. Waters, "Candidate indistinguishability obfuscation and functional encryption for all circuits," in *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, 2013, pp. 40–49. [Online]. Available: https://doi.org/10.1109/FOCS.2013.13

[4] S. Garg, C. Gentry, and S. Halevi, "Candidate multilinear maps from ideal lattices," in *Proc. of EUROCRYPT*, ser. LNCS, vol. 7881. Springer, 2013, pp. 1–17.

[5] J. Coron, T. Lepoint, and M. Tibouchi, "Practical multilinear maps over the integers," in *Advances in Cryptology - CRYPTO 2013*, 2013, pp. 476–493.

[6] C. Gentry, S. Gorbunov, and S. Halevi, "Graph-induced multilinear maps from lattices," in *Theory of Cryptography Conference*. Springer, 2015, pp. 498–527.

[7] P. Ananth, D. Gupta, Y. Ishai, and A. Sahai, "Optimizing obfuscation: Avoiding barrington's theorem," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2014, pp. 646–658.

[8] E. Miles, A. Sahai, and M. Weiss, "Protecting obfuscation against arithmetic attacks." *IACR Cryptology ePrint Archive*, vol. 2014, p. 878, 2014.

[9] B. Barak, S. Garg, Y. T. Kalai, O. Paneth, and A. Sahai, "Protecting obfuscation against algebraic attacks," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2014, pp. 221–238.

[10] S. Badrinarayanan, E. Miles, A. Sahai, and M. Zhandry, "Post-zeroizing obfuscation: New mathematical tools, and the case of evasive circuits," in *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II*, 2016, pp. 764–791. [Online]. Available: https://doi.org/10.1007/978-3-662-49896-5_27

[11] S. Garg, E. Miles, P. Mukherjee, A. Sahai, A. Srinivasan, and M. Zhandry, "Secure obfuscation in a weak multilinear map model," in *Theory of Cryptography - 14th International Conference, TCC 2016-B, Beijing, China, October 31 - November 3, 2016, Proceedings, Part II*, 2016, pp. 241–268. [Online]. Available: https://doi.org/10.1007/978-3-662-53644-5_10

[12] R. Goyal, V. Koppula, and B. Waters, "Lockable obfuscation," in *Foundations of Computer Science (FOCS), 2017 IEEE 58th Annual Symposium on*. IEEE, 2017, pp. 612–621.

[13] D. Wichs and G. Zirdelis, "Obfuscating compute-and-compare programs under lwe," in *Foundations of Computer Science (FOCS), 2017 IEEE 58th Annual Symposium on*. IEEE, 2017, pp. 600–611.

[14] S. Halevi, T. Halevi, V. Shoup, and N. Stephens-Davidowitz, "Implementing bp-obfuscation using graph-induced encoding," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, 2017, pp. 783–798. [Online]. Available: http://doi.acm.org/10.1145/3133956.3133976

[15] J. Coron, M. S. Lee, T. Lepoint, and M. Tibouchi, "Zeroizing attacks on indistinguishability obfuscation over CLT13," in *Public-Key Cryptography - PKC 2017 - 20th IACR International Conference on Practice and Theory in Public-Key Cryptography, Amsterdam, The Netherlands, March 28-31, 2017, Proceedings, Part I*, 2017, pp. 41–58. [Online]. Available: https://doi.org/10.1007/978-3-662-54365-8_3

[16] J. Coron, C. Gentry, S. Halevi, T. Lepoint, H. K. Maji, E. Miles, M. Raykova, A. Sahai, and M. Tibouchi, "Zeroizing without low-level zeroes: New MMAP attacks and their limitations," in *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I*, 2015, pp. 247–266. [Online]. Available: https://doi.org/10.1007/978-3-662-47989-6_12

[17] E. Miles, A. Sahai, and M. Zhandry, "Annihilation attacks for multilinear maps: Cryptanalysis of indistinguishability obfuscation over GGH13," in *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part II*, 2016, pp. 629–658. [Online]. Available: https://doi.org/10.1007/978-3-662-53008-5_22

[18] J. H. Cheon, M. Hhan, J. Kim, and C. Lee, "Cryptanalyses of branching program obfuscations over ggh13 multilinear map from ntru attack," Cryptology ePrint Archive, Report 2018/408, 2018, https://eprint.iacr.org/2018/408, to appear in Crypto 2018.

[19] Y. Chen, C. Gentry, and S. Halevi, "Cryptanalyses of candidate branching program obfuscators," in *Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part III*, 2017, pp. 278–307. [Online]. Available: https://doi.org/10.1007/978-3-319-56617-7_10

[20] Y. Chen, V. Vaikuntanathan, and H. Wee, "Ggh15 beyond permutation branching programs: Proofs, attacks, and candidates," Cryptology ePrint Archive, Report 2018/360, 2018, https://eprint.iacr.org/2018/360.

[21] B. Barak, N. Bitansky, R. Canetti, Y. T. Kalai, O. Paneth, and A. Sahai, "Obfuscation for evasive functions," in *Theory of Cryptography Conference*. Springer, 2014, pp. 26–51.

[22] Z. Brakerski and G. N. Rothblum, "Obfuscating conjunctions," *Journal of Cryptology*, vol. 30, no. 1, pp. 289–320, 2017.

[23] D. Micciancio and C. Peikert, "Trapdoors for lattices: Simpler, tighter, faster, smaller," in *Proc. of EUROCRYPT*, ser. LNCS, vol. 7237. Springer, 2012, pp. 700–718.