Limitation of the HHSS Obfuscation: Lattice based Distinguishing Attack

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

Seoul National University (SNU), Republic of Korea

Abstract. Indistinguishability Obfuscation (iO) is a hopeful tool which obfuscates a program with the least leakage, and produces various applications including functional encryption. Recently, a state-of-the-art obfuscator implementation underlying the branching program matrix, HHSS, has been suggested by Halevi *et. al.* in ACM-CCS'17.
In this work, we describe the first attack algorithm which can be applied to HHSS obfuscation depending on the dimension of the branching program matrix. When two matrix branching programs and an obfuscated program using HHSS obfuscation are given, we can distinguish which branching program was used to make an obfuscated program.
Our attack uses a left kernel of the product of branching program matrices. If we obtain a short vector in the left kernel, we manipulate the result of the zerotesting procedure because HHSS obfuscation removes a special setting called 'scalar bundling' in the initialization step for its efficiency. More precisely, the zerotesting procedure exposes the left kernel of the product of branching program matrices, so we can use the property employing a lattice reduction algorithm on the left kernel. Indeed, we find a short vector what we want using a lattice reduction algorithm. As a result, we can find a vector what we want in the complexity $2^{O(\frac{d}{B-\epsilon})}$, where d is the dimension of the branching program matrices and a gap parameter B and a real value ϵ are given. For example, we can find a short vector applying the LLL algorithm for the current parameter proposed by HHSS implementation with $d = 100$. It takes less than a second with the precomputation of the evaluation of the obfuscated program.
Keywords: Cryptanalysis, Indistinguishability Obfuscation, Matrix Branching Program, Multilinear Maps

1 Introduction

The program obfuscator, also called software obfuscator, is a compiler which takes a program P as an input, and outputs a new program $\mathcal{O}(P)$ which computes the same function as P but it is infeasible to extract information of P. By obfuscating, any adversary cannot reverse engineer the program.

Because of this dream property, the obfuscation has been extensively researched. The indistinguishability obfuscation (iO), also known as best-possible obfuscation [GR07], is the obfuscation which makes the program leak as little information as any other program with the same functionality and a similar size.

The security assumption of the iO, *indistinguishability*, is that any adversary cannot computationally distinguish the obfuscated programs $i\mathcal{O}(BP_0)$ and $i\mathcal{O}(BP_1)$ for the same functionally equivalent and the same size programs BP_0 and BP_1 in a polynomial time.

In FOCS 2013, Garg *et. al.* firstly suggested a plausible candidate, GGHRSW, of *iO* [GGH+13b] based on the matrix branching program (BP), which represent the program by several matrices and a input function and evaluate by multiplying matrices, and the cryptographic multilinear map [GGH13a, CLT13, GGH15].

Recently, GGH15 based BP obfuscation has been in the spotlight because there are schemes which claim provable security under the LWE assumption [GKW17, WZ17]. In particular, Halevi *et. al.* presented a new construction of iO and were successfully implemented their construction [HHSS17]. Their construction does not use scalar bundling for the efficiency of the implementation, and then only obfuscates the read-once branching programs. Therefore, HHSS obfuscation can obfuscate more general programs compared to [LMA+16], which only obfuscates the point functions.

Branching program based *iO* and HHSS implementation. A matrix branching program with dimension d and length ℓ for t-bit input is the set of integral matrix $BP = \{\mathbf{M}_{i,b}\}_{1\le i\le \ell,\ b\in\{0,1\}} \subset \mathbb{Z}^{d\times d}$ with an input function $\mathsf{inp} : [\ell] \to [t]$. On a input $x \in \{0,1\}^t$, BP can be evaluated as follows where x_i is the i-th bit of x:

$$BP(x) = \begin{cases} 0 & \text{if } \prod_{i=1}^{l} \mathbf{M}_{i,x_{\mathsf{inp}(i)}} = \mathbf{0} \\ 1 & \text{otherwise.} \end{cases}$$

The first candidate of *iO* [GGH+13b] is based on the branching program and cryptographic multilinear map ENC. Briefly, it gives a zerotesting procedure, which returns 1 if the top level encodings is zero; otherwise return 0, and a set of encoded matrices

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

where $\mathbf{R}_i$'s are random invertible matrices except $\mathbf{R}_0 = \mathbf{R}_\ell = \mathbf{I}_d$ and $\alpha_{i,b}$'s are scalar bundling elements. Since the multilinear maps allow homomorphic evaluation, one can evaluate the $iO(BP)$ on input $x \in \{0,1\}^t$. Moreover, by checking whether or not the each entry is zero with zerotesting procedure, one can guarantee the functionality is preserved.

In the case of HHSS implementation, to increase efficiency, the $\alpha_{i,b}$ values are unified to all ones and the input function is set as the identity function. These branching programs are called read-once programs. The encoding of HHSS is a $m \times m$ matrix and it is of the form of [GGH15] multilinear map:

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

where the $\mathbf{A}_i$ and $\mathbf{E}_i$ are $n \times m$ random matrices and $n \times m$ small error matrices, respectively, and q is an integer. In addition, the evaluation of the encoded matrices on input $x \in \{0,1\}^\ell$ has the following form:

$$\prod_{i=1}^{\ell} \mathbf{M}_{i,x_i} \cdot \mathbf{A}_\ell + \mathbf{E}.$$

Our Technique. We observe that the size of the evaluation value heavily relies on the matrix $\mathbf{M}_{i,x_i}$. Namely, the size become small when the matrix is zero, and it would be large if not. Moreover, the plaintext matrices are revealed with errors in the evaluation value as the left-multiplied matrix. Since we can assume that we know $\mathbf{M}_{i,b}$, we can compute the left kernel of $\mathbf{M}_{i,b}$ and a short vector in the kernel by using lattice algorithms. If the dimension d of plain matrix branching program is small, we can find such a short vector. Then, the multiplication of a short left kernel vector and the evaluation value would be small if the given obfuscated program was generated form $\mathbf{M}_{i,b}$. Otherwise, the product would become a large value.

Our Contribution. In this work, we propose a noteworthy distinguishing attack on HHSS implementation. From this algorithm, we point out that the structural weakness of HHSS obfuscation due to omitting the scalar bundling randomization. Our distinguishing attack for HHSS implementation relies on the dimension of plain branching program matrices.

More precisely, we suggest a probabilistic algorithm to distinguish whether b is 0 or 1, given that two functionally equivalent branching programs BP_0 and BP_1 and obfuscated program $iO(BP_b)$ from HHSS implementation. In other words, we show that HHSS implementation does not achieve indistinguishability. Let B be the gap between $\log_2 q$ and the size of error of top level encodings and ϵ be a real number greater than 1 which is related to probability of success of the attack. Our algorithm then runs in $2^{O(\frac{d}{B-\epsilon})}$ and the success probability of distinguishing attack is at least $1/2 - 1/2^{\epsilon+1}$. According to our attack, if d is smaller than a security parameter $\lambda \cdot (B-\epsilon)$, we show that HHSS implementation does not satisfy the indistinguishability with the security parameter λ.

However, our attack implies that these programs cannot be obfuscated using HHSS. Incidentally, our results also naturally increase the size of the obfuscated matrices and reduce the efficiency of the HHSS implementation. In other words, the class that HHSS can practical obfuscate is limited.

Comparison to concurrent and independent work. The concurrent and independent work of Chen, Vaikuntanathan, and Wee proposed another polynomial time attack on HHSS obfuscation, so called rank attack, in the paper [CVW18]. A little more precisely, their main idea is to construct a special matrix from several top level encodings of zero for a given obfuscated program. The rank of the matrix depends on the plain branching program. In other words, given a branching program P and an obfuscated program O with same functionality as P, by computing the rank, one can distinguish whether O is an obfuscated program of P or not. This attack implies the traditional general purpose iO scheme over GGH15 graded encoding scheme does not satisfy indistinguishability even though the obfuscator includes various safeguards such as Kilian-style randomization, scalar bundling, and diagonal padding.

Compared to our attack, the rank attack requires several top level encodings of zero, so it is not applicable to obfuscated BP programs that anyone hardly obtain encodings of zero such as an evasive function or a point function On the other hand, our attack affects HHSS obfuscation of such branching programs which do not even give encodings of zero. However, out attack is only applicable to an obfuscator with Kilian-style randomization and diagonal padding. In summary, we demonstrate that our attack range is broader than the attackable program class in HHSS obfuscation, and that the attack range of Chen et al. is broader than the attackable obfuscation scheme class.

Organization. The preliminaries related to obfuscation are presented in Section 2. The scheme description of HHSS obfuscation is discussed in Section 3. Next, our distinguishing algorithm with its experimental example and a new parameter selection are given in Section 4 and finally Section 5 gives the conclusion.

2 Preliminaries

Notation. For an integer $q \geq 2$, $\mathbb{Z}_q$ 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 $\|\mathbf{v}\|_{op} \leq \|\mathbf{v}\|_2$ 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 $i\mathcal{O}$ 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,*

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

- *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* negl *such that*

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

Security of *iO*. Let $\mathcal{B}$ and $\mathcal{B}'$ be given two matrix branching programs whose outputs are the same for all inputs x. Then, the security goal of *iO* is to make sure that no PPT distinguisher $\mathcal{D}$ can distinguish whether an obfuscated program comes from $\mathcal{B}$ or $\mathcal{B}'$.

Moreover, we say "$i\mathcal{O}$ does not have indistinguishability" if there exists a PPT distinguisher $\mathcal{D}$ which can determine that the given obfuscated program $i\mathcal{O}$ is an obfuscation of $\mathcal{B}$ or an obfuscation of $\mathcal{B}'$.

2.1 Lattice

Lattice. A lattice $\mathcal{L} \subset \mathbb{R}^m$ is the discrete set generated by a set of linearly independent column vectors of $\mathbb{R}^m$. For given n linearly independent column vectors $\mathbf{b}_1, \mathbf{b}_2, \cdots, \mathbf{b}_n \in \mathbb{R}^m$, the lattice generated by the set $\mathbf{B} = \{\mathbf{b}_1, \mathbf{b}_2, \cdots, \mathbf{b}_n\}$ with $n \le m$ is the set $\mathcal{L} = \left\{ \sum_{i=1}^{n} a_i \cdot \mathbf{b}_i : a_i \in \mathbb{Z} \right\}$ of integer linear combinations of the $\mathbf{b}_i$, and denoted by $\mathcal{L}(\mathbf{B})$. We call m the dimension and n the rank of lattice $\mathcal{L}$. The determinant $\det(\mathcal{L})$ of the lattice $\mathcal{L}$ is defined as $\sqrt{\det(\mathbf{B}^T \cdot \mathbf{B})}$ for any basis matrix $\mathbf{B}$ of $\mathcal{L}$.

Lattice reduction algorithm. Lattice reduction algorithms are usually used to find a short vector in a lattice. LLL algorithm and BKZ algorithm are well known examples of lattice reduction algorithm, and details of these algorithms are described in [LLL82, NS06, Ngu11, HPS11]. Two algorithms return a reduced basis where the first vector is relatively short in the lattice. In this paper, lattice reduction algorithms are only used to a find short vector of a given lattice. When employing lattice reduction algorithm on an n-dimensional lattice $\mathcal{L}$, the shortest output of the algorithm, say $\mathbf{b}_1$, satisfies a condition the form: $\|\mathbf{b}_1\| \le \delta^n \cdot \det(\mathcal{L})^{1/n}$ for some constant δ. We call δ the *root Hermite factor*.

3 HHSS Indistinguishability Obfuscation

Since the advent of [GGH+13b], the construction of candidate *iO* is largely composed of two steps; matrix randomization and encoding using multilinear map. To instantiate obfuscation and other various applications, three candidates of cryptographic multilinear maps are suggested [GGH13a, CLT13, GGH15].

Halevi *et. al.* proposed and implemented an obfuscation [HHSS17] based on the branching program and cryptographic multilinear map to achieve indistinguishability obfuscation. They applied several randomization steps on the matrix branching program and encoded the randomized matrices by GGH15 multilinear map.

In this section, we briefly review GGH15 multilinear map and HHSS obfuscation construction. For more details, refer to [HHSS17].

GGH15 multilinear map, which is a graph-induced cryptographic multilinear map, 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 GGH15 multilinear map. In particular, we only consider a directed graph G which consists of two chains of length ℓ with a common *source* vertex 0 and a common *sink* vertext ℓ. 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

$$0 \to 1 \to 2 \to \cdots (\ell-1) \to \ell \ \text{ and } \ 0 \to 1' \to 2' \to \cdots (\ell-1)' \to \ell$$

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

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 τ_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 τ_u [MP12].

In HHSS obfuscation, the public parameters of GGH15 multilinear map 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

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

- 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 $\|\mathbf{A}_0 \cdot \mathbf{C}\|$ for a given encoding $\mathbf{C}$ of $\mathbf{M}$ with respect to edge $0 \to \ell$. Specially, if $\|\mathbf{A}_0 \cdot \mathbf{C}\| \leq q/2^{10}$, $\mathbf{C}$ is an encoding matrix of $\mathbf{M} = \mathbf{0}_n$.

3.1 Construction of HHSS Obfuscation

Randomizing Branching Program. As noted above, HHSS obfuscation only support the read-once branching programs since they remove *scalar bundlings* [1]. A matrix branching program of read-once program for ℓ-bit input is ℓ pairs of $d \times d$ matrices

$$\mathcal{B} = \{(\mathbf{M}_{1,0}, \mathbf{M}_{1,1}), (\mathbf{M}_{2,0}, \mathbf{M}_{2,1}), \cdots, (\mathbf{M}_{\ell,0}, \mathbf{M}_{\ell,1})\}$$

such that

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

where x_i is the i-th bit of input x. In other words, a function inp is the identity.

Halevi *et. al.* use two randomization steps called higher-dimensional embedding and Kilian-style randomization upon a given read-once branching program $\mathcal{B}$. 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 [HHSS17], they picked $d' = \lceil \sqrt{\lambda/2} \rceil$. After higher-dimensional embedding, they apply Kilian-style randomization 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

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

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.

[1] 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.

Halevi *et. al.* employ an additional structure which is called the *dummy program* for the zerotesting procedure. It consists of ℓ 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

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

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 GGH15 Multilinear map. We describe how to employ the GGH15 multilinear map to encode randomized matrices.

After randomization steps, we recall the setting of GGH15 multilinear map. 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 ℓ, 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 *safeguard*, to make attacks harder since its security is suspect. Safeguard uses a Kilian style randomization again to the output of encodings. Applying the safeguard step, we have

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

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:

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

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 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 Error term largely depends on the term $\mathbf{E}_1 \cdot \prod_{i=2}^{\ell} \mathbf{C}_2$ since the size of matrix $\mathbf{C}_i$ is larger than that of other $\mathbf{M}_i$, $\mathbf{S}_{j-1}$ and noise term $\mathbf{E}_j$ for all i, j. Therefore, we can speculate the norm 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 σ_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 [HHSS17] or Appendix.

Halevi *et.al.* designed a zerotesting procedure by estimating $\|\mathbf{A}_0 \cdot \mathbf{C}\|$. Namely, the obfuscated program $i\mathcal{O}(BP)(x)$ outputs 0 when $\|\mathbf{A}_0 \cdot \mathbf{C}\| \leq q/2^{10}$. Similarly, $i\mathcal{O}(BP)(x)$ 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.

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

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

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

Remark. To reduce the size of $i\mathcal{O}(BP)$, Halevi *et. al.* proposed a special encoding for sink ℓ. 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 $\ell - 1 \rightarrow \ell$, 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 $\hat{\mathbf{C}} = \mathbf{P}_{\ell-1} \cdot \mathbf{C}_\ell$.

4 Cryptanalysis of the HHSS Implementation

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 assume that these matrices are binary matrices since the given programs in HHSS implementation are all binary. 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.

4.1 Description of Our Attack.

Let $\mathcal{B}$ and $\mathcal{B}'$ be two read-once matrix branching programs for ℓ-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 $i\mathcal{O}$ encoded by GH15 multilinear map, but we do not know whether or not $i\mathcal{O}$ is an obfuscation whether $i\mathcal{O}$ is an obfuscation of the branching program $\mathcal{B}$ or $\mathcal{B}'$. We want to determine regardless of whether an obfuscated program $i\mathcal{O}$ comes from $\mathcal{B}$ or $\mathcal{B}'$.

Suppose we have public matrix $\mathbf{A}_0$ of GGH15 multilinear map and following matrices.

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

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

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 GGH15 multilinear map, 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 $f_{\mathcal{B}}(x) = 0$ 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. Also, there is a nonzero vector $\mathbf{v} \in \mathbb{Z}^d$ such that $\mathbf{v} \cdot \mathbf{M}_{1,1} = \mathbf{0}_d$. *i.e.*, the left kernel of a matrix $\mathbf{M}_{1,1}$ is not trivial.[3] 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 $f_{\mathcal{B}}(x) = 1$ and $f_{\mathcal{B}'}(x) = 1$. 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. We define a new real value $\epsilon \geq 1$ in order to clarify how much small the term $\hat{\mathbf{v}} \cdot \mathsf{Error} \pmod q$ is. The ϵ is exploited to predict the probability to distinguish program later. 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

$$\mathcal{D}(i\mathcal{O}) = \begin{cases} \mathcal{B} & \text{if } \|\hat{\mathbf{v}} \cdot (\mathbf{A}_0 \cdot \hat{\mathbf{C}})\|_{op} \leq q/2^{\epsilon} \text{ for some integer } \epsilon. \\ \mathcal{B}' & \text{othersiwe.} \end{cases}$$

[3] Sometimes it is called the ‘orthogonal lattice’ of the column lattice generated by a matrix $\mathbf{M}_{1,1}$.

Extending the attack. Some programs do not have 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. Moreover, the determinant of a lattice generated by $\mathbf{M}^{(1)}$ is no more than $\sqrt{2d}^d$ because the entries of a new matrix are also binary. Thus, we conclude our attack can be applied to HHSS obfuscation.

In summary, our attack is explicitly described as follows:

Algorithm 1 Left Kernel Attack (Simple case)

```
Input: $q, \epsilon$, $\mathbf{A}_0$, $\mathcal{B} = \{\mathbf{M}_{i,b}\}_{i \in [\ell], b \in \{0,1\}}$, $i\mathcal{O}(\mathcal{B}') = \{\hat{\mathbf{C}}_{i,b}\}_{i \in [\ell], b \in \{0,1\}}$
Output: True if $\mathcal{B} = \mathcal{B}'$, and False otherwise
 1: compute the left kernel $\mathbf{K}$ of $\mathbf{M}^{(1)}$ and its basis $\mathbf{B}$
 2: find a short left kernel vector $\mathbf{v} \in \mathbf{K}$ by using a lattice reduction algorithm
 3: $\hat{\mathbf{v}} = (\mathbf{v} || \mathbf{0}) \in \mathbb{Z}^{d+d'}$ with $d'$-dimension vector $\mathbf{0}$
 4: compute $\hat{\mathbf{C}} = \prod_{i=1}^{\ell} \hat{\mathbf{C}}_{i,x_i}$ for $x_1 = 0$
 5: compute $z = \hat{\mathbf{v}} \cdot \mathbf{A}_0 \cdot \hat{\mathbf{C}}$
 6: return True if $|z| \leq q/2^{\epsilon}$; otherwise return False.
```

Toy Example. Now, we give two simple equivalent branching programs and the results of our attack. Two branching programs $\mathcal{B}, \mathcal{B}'$, which satisfy $f_{\mathcal{B}}(x) = f_{\mathcal{B}'}(x) = 0$ for $x = 01$ and $f_{\mathcal{B}}(x) = f_{\mathcal{B}'}(x) = 1$ otherwise, are given as follows:

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

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

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

respectively. If we set the security parameter $\lambda = 80$ then the embedding dimension $d' = 5$[4] 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:

$$\mathbf{A}_0 \cdot \hat{\mathbf{C}}_{0,1} \cdot \hat{\mathbf{C}}_{1,0} = \begin{pmatrix} -10884489 \\ 6341880489273 \\ -2809809111564 \\ -9752397591639 \\ -3576418758634 \\ -7724278907092 \\ 11995182501421 \end{pmatrix}, \ \mathbf{A}_0 \cdot \hat{\mathbf{D}}_{0,1} \cdot \hat{\mathbf{D}}_{1,0} = \begin{pmatrix} -281520684456 \\ -281549139175 \\ -410308959247 \\ -15011968960796 \\ -5708322256588 \\ -2185335626767 \\ -593005703340 \end{pmatrix}.$$

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

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

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

4.2 Analysis of the Left Kernel Attack

Let $\mathbf{M}^{(1)}$ be the matrix defined as above and a lattice $\mathcal{L}^{(1)}$ generated by column vectors of $\mathbf{M}^{(1)}$. We denote a lattice ${\mathcal{L}^{(1)}}^{\perp}$ by an orthogonal lattice of a lattice $\mathcal{L}^{(1)}$. Then, ${\mathcal{L}^{(1)}}^{\perp}$ is the same as a nontrivial left kernel of a matrix $\mathbf{M}^{(1)}$. Moreover, ${\mathcal{L}^{(1)}}^{\perp}$ has rank d with high probability, and $\det {\mathcal{L}^{(1)}}^{\perp} \leq \det \mathcal{L}^{(1)} = \sqrt{2d}^{d}$. On the lattice $\mathcal{L}^{(1)}$, if we can find a short vector $\mathbf{v}' = (\mathbf{u}||\mathbf{w})$ and compute $(\mathbf{u}||\mathbf{0}) \cdot (\mathbf{A}_0 \cdot \hat{\mathbf{C}}_0) + (\mathbf{w}||\mathbf{0}) \cdot (\mathbf{A}_0 \cdot \hat{\mathbf{C}}_1)$, where $\hat{\mathbf{C}}_0$ and $\hat{\mathbf{C}}_1$ are encodings of $\mathbf{M}_{1,1} \cdot \prod_{i=2}^{\ell} \mathbf{M}_{i,x_1}$ and $\mathbf{M}_{1,0} \cdot \prod_{i=2}^{\ell} \mathbf{M}_{i,x_1}$ with respect to the path $0 \to \ell$, then we know which program is used to make the obfuscation $i\mathcal{O}$.

Now, we try to analyze an operator norm $\|(\mathbf{u}||\mathbf{0}) \cdot \mathsf{Error}_1 + (\mathbf{w}||\mathbf{0}) \cdot \mathsf{Error}_2\|_{op}$ to find an upper bound of a vector $\mathbf{v}'$ what we need. Let Error_1 and Error_2 be error matrices in the encodings $\hat{\mathbf{C}}_0$ and $\hat{\mathbf{C}}_1$ respectively, then, $\|\mathsf{Error}_j\|_{op}$ $(j = 1, 2)$ is bounded by $2^7 \cdot \sigma_x^{\ell-1} \cdot m^{\ell/2} \cdot 2^{\ell-1} =: y$.

Consider a $2n$-dimensional vector

$$\mathbf{V} = (\mathbf{u}||\mathbf{0}||\mathbf{w}||\mathbf{0}) \cdot \begin{pmatrix} \mathsf{Error}_1 \\ \mathsf{Error}_2 \end{pmatrix}.$$

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

Then, $\|\mathbf{V}\|_{op} = \|(\mathbf{u}||\mathbf{0}) \cdot \mathsf{Error}_1 + (\mathbf{w}||\mathbf{0}) \cdot \mathsf{Error}_2\|_{op}$. Moreover, the following inequality holds: $\|(\mathbf{u}||\mathbf{0}||\mathbf{w}||\mathbf{0})\|_{op} \leq \|\mathbf{v}'\|_{op} \leq \|\mathbf{v}'\|_2$ and $\|(\mathsf{Error}_1||\mathsf{Error}_2)^T\|_{op}$ is bounded by $\sqrt{2}y$.

Therefore, we conclude

$$\begin{aligned}\|(\mathbf{u}||\mathbf{0}) \cdot \mathsf{Error}_1 + (\mathbf{w}||\mathbf{0}) \cdot \mathsf{Error}_2\|_{op} &\leq \|(\mathbf{u}||\mathbf{0}||\mathbf{w}||\mathbf{0})\|_{op} \cdot \left\| \begin{pmatrix} \mathsf{Error}_1 \\ \mathsf{Error}_2 \end{pmatrix} \right\|_{op} \\ &\leq \sqrt{2} \cdot y \cdot \|\mathbf{v}'\|_{op} \leq \sqrt{2} \cdot y \cdot \|\mathbf{v}'\|_2.\end{aligned}$$

Then, a short vector $\mathbf{v}' = (\mathbf{u}||\mathbf{w})$ with $\mathbf{u}, \mathbf{w} \in \mathbb{Z}^d$ should satisfy

$$\|\mathbf{v}'\|_2 \leq \delta^d \cdot (\det \mathcal{L}^{(1)^\perp})^{\frac{1}{d}} \leq \frac{q}{2^\epsilon \cdot \sqrt{2}y} =: 2^B/2^\epsilon,$$

where δ is the root Hermite factor of a lattice reduction algorithm and $q/\sqrt{2}y$ is replaced by 2^B. We remark B indicates that a gap between underlying modulus size $\log_2 q$ and the size of Error. Note that $\delta^d \cdot (\det \mathcal{L}^{(1)^\perp})^{\frac{1}{d}} \leq \delta^{2d} \cdot (\sqrt{2d}^d)^{\frac{1}{d}}$ because $\det \mathcal{L}^{(1)^\perp} \leq \sqrt{2d}^d$.

In summary, we need to check whether the following inequality holds

$$\delta^d \cdot \sqrt{2d} \leq \frac{q}{2^\epsilon \cdot \sqrt{2}y}.$$

Success Probability. The probability that the algorithm distinguishes the programs $\mathbf{M}$ and $\mathbf{N}$ from which $i\mathcal{O}(\mathbf{M})$ obfuscated is as follows. Let $\mathbf{v}$ and $\mathbf{w}$ be left kernel vectors of $(\mathbf{M}_{1,0}||\mathbf{M}_{1,1})^T$ and $(\mathbf{N}_{1,0}||\mathbf{N}_{1,1})^T$, respectively. Then, $\mathbf{v} \cdot (\mathbf{A}_0 \cdot \hat{\mathbf{C}})$ is unconditionally smaller than $q/2^\epsilon$. That is, $\Pr[\mathcal{D}(\mathbf{M}, i\mathcal{O}(\mathbf{M})) = \mathbf{M}] = 1/2$. *i.e.*, a distinguisher $\mathcal{D}$ (Algorithm 1) can output $\mathbf{M}$ with probability $1/2$ when $i\mathcal{O}(\mathbf{M})$ is given as an input. On the other hand, assuming that $\mathbf{w} \cdot (\mathbf{A}_0 \cdot \hat{\mathbf{C}})$ is a random value, the probability of $\|\mathbf{w} \cdot (\mathbf{A}_0 \cdot \hat{\mathbf{C}})\|_{op} \leq q/2^\epsilon$ is smaller than $(1/2^\epsilon)^n$ because all entries must be smaller than $q/2^\epsilon$. It implies that $\Pr[\mathcal{D}(\mathbf{N}, i\mathcal{O}(\mathbf{M})) = \mathbf{N})] = (1/2^{n\cdot\epsilon+1})$. In summary, we have the advantage of distinguishing by the probability $|\Pr[\mathcal{D}(\mathbf{M}, i\mathcal{O}(\mathbf{M})) = \mathbf{M}] - \Pr[\mathcal{D}(\mathbf{N}, i\mathcal{O}(\mathbf{M})) = \mathbf{N}]| = 1/2 - 1/2^{n\cdot\epsilon+1}$, which is overwhelming probability on n.

Asymptotic analysis. We give an asymptotic complexity of our attack with respect to d. In order to obtain such a vector $\mathbf{v}'$, the lattice reduction algorithm like LLL algorithm and BKZ algorithm [LLL82, HPS11] is required. When the term $B - \epsilon$ is set as $\Omega(d)$, it can be recovered with the LLL algorithm with polynomial time in n. In the case of $B - \epsilon = \Omega(d/\beta)$, it can be found with BKZ algorithm with block size β and it runs in $2^{O(\beta)}$. In other words, our algorithm runs in $2^{\frac{d}{B-\epsilon}}$. Therefore, to resist our attack, d is set as $O(\lambda \cdot (B - \epsilon))$.

Attack for concrete Parameters. We will now use the lattice reduction algorithm with the root Hermite factor δ to show how *iO* does not have indistinguishability for concrete parameters.

In the paper [HHSS17], they set parameters $d = 100$, $n = 105$ and commented $\delta \approx 1.006$ for a security parameter $\lambda = 80$. Moreover, they proposed parameters $\ell, \sigma_x, \log_2 q$ and m which are used to implement an *iO* using GGH15 mmap, and we describe them in Table 1. As stated above, maximum of the size $\log_2 \|\mathbf{v}'\|_2$ must be less than $B - \epsilon = \log_2 q - \log_2 y - \epsilon - 1/2$. The details of the bound B of $\max \log_2 \|\mathbf{v}'\|_2$ are in Table 2.

ℓ	5	8	10	12	14	17	20
σ_x	$2^{18.0}$	$2^{18.3}$	$2^{18.4}$	$2^{18.6}$	$2^{18.7}$	$2^{18.8}$	$2^{18.9}$
$\log_2 q$	133	219	261	322	382	458	542
m	3352	5621	6730	8339	9923	11928	14145

Table 1: Parameters in [HHSS17] when security $\lambda = 80$, $d = 100$

ℓ	5	8	10	12	14	17	20
$\log_2 q$	133	219	261	322	382	458	542
$\log_2 y$	113	192	246	301	357	439	523
$B - \epsilon$	18	25	13	19	23	17	17

Table 2: Bound B of a target vector when security $\lambda = 80$, $d = 105$, $\epsilon = 1.5$

From the above table, if we can find a short vector $\mathbf{v}'$ such that

$$\|\mathbf{v}'\|_2 \leq \delta^d \cdot \sqrt{2d} \leq 2^{B-\epsilon}, \tag{2}$$

then we can always distinguish a program $i\mathcal{O}$ which comes from $\mathcal{B}$ or $\mathcal{B}'$. Indeed, $\delta^d \cdot (\sqrt{2d}^d)^{\frac{1}{d}} \approx 25.7 \ll 2^{13}$ since the root Hermite factor factor $\delta \approx 1.006$, $B - \epsilon = 13$, and $d = 100$. In conclusion, matrix branching program obfuscation using GGH15 multilinear map does not have indistinguishability using a lattice reduction algorithm with the root Hermite factor $\delta \approx 1.006$ when two *iO* programs are given. Moreover, its implementation proposed in [HHSS17] is not secure when employing current parameters.

4.3 Experiments

In this section, we give specific experimental results of the attack. Halevi *et. al.* provide some examples of branching program and codes for generating random branching programs in https://github.com/shaih/BPobfus.

Table 3 shows that the time of the obfuscator and the attack for given parameters of HHSS implementation. The dimension d of the matrix branching programs is 100. ℓ denotes the length of branching programs, and m the dimension of encoded matrices. The evaluation time (step 4 of the left kernel attack) is assumed precomputed in the attack time on the table. Our attack is carried out in a second for the given parameter setting with preprocessed evaluation matrices.

We remark that short left kernel vectors $\mathbf{v}$ satisfying $\|\mathbf{v}\|_\infty \leq 10$ are found by using LLL algorithm for 100 dimensional randomly generated matrix branching programs, which is used in the experiments of [HHSS17]. Moreover, all of the example branching programs given in Github have a binary left kernel vectors.

According to the experiments, the product $\mathbf{v} \cdot (\mathbf{A}_0 \cdot \hat{\mathbf{C}})$ is far smaller than $2^{-10} \times q$ for the short left kernel vector $\mathbf{v}$ from the corresponding plain matrix branching program, whereas it seems a random value modulus q for the left kernel vector from the different plain matrix branching program.

The results of the experiments are given in Table 4. The given examples are the obfuscation of length $\ell = 5$ randomly generated matrix branching programs and we choose $\epsilon = 10$ and use LLL lattice reduction algorithm [**?**]. We note that the attack should employ the other lattice reduction algorithms to distinguish larger branching program matrices.

In all experiments, the dominant time of the left kernel attack of the suggested parameters is the evaluation time of the given obfuscated program. Short vectors of the left kernel of the given plain branching program matrix with the size $(\leq 400) \times (\leq 200)$ are computed in only few seconds by using LLL algorithm with NTL library. Computing the product of the evaluation matrix and the short left kernel vector (modulo q) is also very fast.

We tested all experiments on a server with Intel Xeon Core E5-2620 running at 2.10 GHz processors with 8 threads. We also remark that the obfuscator runs slower than Halevi *et. al.* presented.

ℓ	m	$\log q$	Initialization	Obufscation	Evaluation	Attack
(8×) Intel Xeon CPU E5-2620 2.10GHz						
5	3352	134	75.44	330.95	6.85	0.64
8	5621	220	830.07	3017.01	76.45	0.69
10	6730	262	1967.55	9182.40	147.475	0.68

Table 3: Running time (seconds) of the algorithms (ℓ is the length of progams)

d	m	Initialization	Obufscation	Evaluation	Attack
100	3352	75.87	377.77	7.12	0.67
150	4368	148.80	748.70	12.13	2.53
200	5504	275.36	946.49	19.35	7.23

Table 4: Running time (seconds) of the algorithms (d is the number of states)

4.4 Parameter Suggestion

We present the parameter suggestion for HHSS scheme to achieve 2^λ security for *iO* and to satisfy the functionality/correctness conditions. Basically, as in the HHSS scheme, for $q = \prod_{i=1}^{k} p_i^e$, the parameter e is set to be 3, p_i is ranged from 71 to 181, the standard deviation $\sigma_{\mathbf{z}}$ of $\mathbf{z}$ expressed in **Algorithm 2** [5], σ_z, is assumed to 724, and the dimension of the ciphertext matrix m is set as $\bar{m} + dke$. The remaining parameters $\{B, d, \bar{m}, q, y\}$ are set as follows to achieve 2^λ security against our attack.

- $B, \epsilon \geq 1$, by default, $B = 14.5$ and $\epsilon = 1.5$
- d : the smallest integer satisfying $\delta^d \cdot \sqrt{2d} > 2^{B-\epsilon}$ according to Eq. 2. [6]
- $\bar{m} \geq (\sqrt{\lambda} + 2) \cdot \sqrt{d \cdot \log_2 q}$ according to Eq. 3. [5]
- $\log_2 y = \log_2 \sigma_x \cdot (\ell - 1) + \log_2 m \cdot \ell/2 + \ell + 6$.
- $\log_2 q = \log_2 y + 1/2 + B - \epsilon$ according to Eq. 1.
- $\sigma_x > 2896 \cdot (\sqrt{\bar{m}} + \sqrt{w} + 6)$ according to Eq. 4. [5]

From the above parameter selection, given the security parameter $\lambda = 80$, d should set to be 882. The Table 5 indicates that other parameter values in each length ℓ. The *iO* size means the storage of the set of output of obfuscated matrices

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

5 Conclusion

In this paper, we proposed a new cryptanalysis called left kernel attack. To prevent our attack, the parameters of the obfuscation should be increased. Specifically, the dimension of plaintext matrix and the dimension of encoded matrix would be increased at least five times and at least three times, respectively, than that of the Halevi *et. al.*'s suggestion. This increase results in the storage usage and the time to obfuscate/evaluate the program increase to approximately

[5] Refer to trapdoor sampling in appendix.

[6] Like a paper HHSS17, a root Hermite factor δ is used when $\delta \approx 1.006$ for $\lambda = 80$, $\delta \approx 1.00044$ for $\lambda = 128$, and $\delta \approx 1.0023$ for $\lambda = 256$.

ℓ	5	8	10	12	14	17	20
σ_x	$2^{18.1}$	$2^{18.21}$	$2^{18.26}$	$2^{18.31}$	$2^{18.35}$	$2^{18.40}$	$2^{18.44}$
$\log_2 q$	134	216	271	327	383	467	553
$\bar{m}$	3,763	4,777	5,351	5,878	6,361	7,024	7,644
m	19,639	31,237	39,749	45,568	53,989	62,590	73,794
iO size	94.0GB	670GB	1.75TB	3.39 TB	6.59TB	13.31TB	26.01TB

Table 5: Parameters suggestion to prevent left kernal attack when security parameter $\lambda = 80$, $\epsilon = 1.5$, $B = 14.5$, and $d = 882$

ten times as much. As an illustration, their signature example, which is 20 nibbles (80 bits) and 100 states program obfuscation, would take approximately a year to obfuscate the program. This size increase reduces the range of HHSS indistinguishability obfuscator that can be applied.

All attacks proposed in *iO* area are related to distinguishing between two branching programs instead of finding information leakage in the given obfuscated program. Hence, the problem of extracting secret information such as secret key or plaintext matrices from the obfuscated program remains as an open question. Moreover, even if someone successfully recovers the plaintext matrices, it is difficult to recover the original message that is hidden by Kilian randomization. How to extract some leakage from the specific program obfuscation is also an interesting open question.

References

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

[CVW18] Yilei Chen, Vinod Vaikuntanathan, and Hoeteck Wee. Ggh15 beyond permutation branching programs: Proofs, attacks, and candidates. Cryptology ePrint Archive, Report 2018/360, 2018. https://eprint.iacr.org/2018/360.

[GGH13a] S. Garg, C. Gentry, and S. Halevi. Candidate multilinear maps from ideal lattices. In *Proc. of EUROCRYPT*, volume 7881 of *LNCS*, 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 *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 40–49, 2013.

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

[GKW17] Rishab Goyal, Venkata Koppula, and Brent Waters. Lockable obfuscation. In *Foundations of Computer Science (FOCS), 2017 IEEE 58th Annual Symposium on*, pages 612–621. IEEE, 2017.

[GR07] Shafi Goldwasser and Guy 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*, pages 194–213, 2007.

[HHSS17] 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, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, pages 783–798, 2017.

[HPS11] Guillaume Hanrot, Xavier Pujol, and Damien Stehlé. Analyzing blockwise lattice algorithms using dynamical systems. In *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings*, pages 447–464, 2011.

[LLL82] Arjen Klaas Lenstra, Hendrik Willem Lenstra, and László Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261(4):515–534, 1982.

[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, Vienna, Austria, October 24-28, 2016*, pages 981–992, 2016.

[MP12] D. Micciancio and C. Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *Proc. of EUROCRYPT*, volume 7237 of *LNCS*, pages 700–718. Springer, 2012.

[Ngu11] Phong Q. Nguyen. Lattice reduction algorithms: Theory and practice. In *Advances in Cryptology - EUROCRYPT 2011 - 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings*, pages 2–6, 2011.

[NS06] Phong Nguyen and Damien Stehlé. Lll on the average. *Algorithmic Number Theory*, pages 238–256, 2006.

[WZ17] Daniel Wichs and Giorgos Zirdelis. Obfuscating compute-and-compare programs under lwe. In *Foundations of Computer Science (FOCS), 2017 IEEE 58th Annual Symposium on*, pages 600–611. IEEE, 2017.

A Trapdoor sampling

In this section, we briefly recall the trapdoor-sampling procedure and a related property employed in [HHSS17]. We note that a matrix $\mathbf{A}$ one of the output of the procedure is exploited in encoding step of *iO* initializing. In addition, a trapdoor sampling step would be used to parameter selection. For more details, we refer to the procedure of section 4 [HHSS17].

At the high level, for a given vector $\mathbf{u} \in \mathbb{Z}_q^n$, the trapdoor sampling outputs a pseudo random matrix $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$ with a trapdoor matrix $\mathbf{R}$ and a small vector $\mathbf{x} \in \mathbb{Z}_q^m$ such that $\mathbf{A} \cdot \mathbf{x} = \mathbf{u} \bmod q$. The procedure consists of two steps. First the matrices $\mathbf{A}$ and $\mathbf{R}$ are constructed from "gadget matrix" $\mathbf{G} \in \mathbb{Z}^{n \times w}$, which is explained below. Next employing the matrix $\mathbf{A}$ and $\mathbf{R}$ and target vector $\mathbf{u}$, the small vector $\mathbf{x}$ what the constraint is satisfied is obtained. In order to realize the algorithm, the authors used the discrete Gaussian sampling. For simplicity, we use the notation $\mathbf{u} \leftarrow D_{\mathbb{Z}^m,\mathbf{\Sigma}}$ to sample a vector $\mathbf{u} \in \mathbb{Z}^m$ from discrete Gaussian distribution over $\mathbb{Z}^m$ with a covariance matrix $\mathbf{\Sigma}$. Especially, when the covariance matrix is of the form $\mathbf{\Sigma} = \sigma \cdot \mathbf{I}$, we denote the distribution as $D_{\mathbb{Z}^m,\sigma}$.

Now we describe more details in each step. In [HHSS17], the underlying modulus q is set as $\prod_{i=1}^{k} p_i^e$, where $\{p_i\}$ are small co-prime factors in the range from 71 to 181, e equals to 3, and k is set large enough that $\log_2 q$-bit satisfies the parameter suggestion [HHSS17, Sec 5.].

Next, they define a vector $\boldsymbol{g} = (g_1, \cdots, g_\omega) \in \mathbb{Z}^\omega$ such that $g_1 = 1$ and $g_{i+1} = g_i \cdot p_{\lfloor i/e \rfloor}$ and a gadget matrix $\mathbf{G} = \boldsymbol{g}^T \otimes \mathbf{I}_n \in \mathbb{Z}^{n \times \omega}$ for $\omega = n \cdot k \cdot e$. We recall that if an arbitrary vector $\mathbf{v}$ is given, it is easy to sample a vector $\mathbf{z}$ of small size satisfying $\mathbf{G} \cdot \mathbf{z} = \mathbf{v}$ by the Micciancio-Peikert method [MP12].

Next, they sample an uniform matrices $\mathbf{A}_1 \in \mathbb{Z}_q^{n \times \bar{m}-2n}$, $\mathbf{A}_2 \in \mathbb{Z}_q^{n \times n}$ until $\mathbf{A}_2$ is invertible and matrices of small norm $\mathbf{R}_1 \in D_{\mathbb{Z},4}^{\bar{m}-2n \times \omega}$, $\mathbf{R}_2 \in D_{\mathbb{Z},4}^{n \times \omega}$, then sets $\mathbf{A} = [\mathbf{A}_1 | \mathbf{A}_2 | \mathbf{G} - \mathbf{A}_1 \cdot \mathbf{R}_1 - \mathbf{A}_2 \cdot \mathbf{R}_2]_q$ and $\mathbf{R} = [\mathbf{R}_1 | \mathbf{R}_2 | \mathbf{I}]^T$ so that the matrix $\mathbf{A}$ holds $\mathbf{A} \cdot \mathbf{R} = \mathbf{G} \bmod q$. In order to guarantee the pseudo-randomness of $\mathbf{A}$, it is enough that $[\mathbf{A}_1 \cdot \mathbf{R}_1 + \mathbf{A}_2 \cdot \mathbf{R}_2]_q = \mathbf{A}_2 \cdot (\mathbf{A}_2^{-1} \cdot \mathbf{A}_1 \cdot \mathbf{R}_1 + \mathbf{R}_2)$ is pseudo-random. Under LWE assumption, $(\mathbf{A}_2^{-1} \cdot \mathbf{A}_1, \mathbf{A}_2^{-1} \cdot \mathbf{A}_1 \cdot \mathbf{R}_1 + \mathbf{R}_2)$ has pseudo-randomness when the followings hold:

$$\bar{m} \geq (\sqrt{\lambda} + 2) \cdot \sqrt{n \log q}. \tag{3}$$

Finally, given the matrix $\mathbf{A}$, trapdoor matrix $\mathbf{R}$, and a target vector $\mathbf{u} \in \mathbb{Z}_q^n$, one can sample a small vector $\mathbf{x}$ satisfying $\mathbf{A} \cdot \mathbf{x} = \mathbf{u} \bmod q$. The process is described in **Algorithm 2**. We remark that σ_x must be sufficiently large relative

Algorithm 2 Computing Encrypted Distance Vectors

```
Input: A, R, and u
Output: x
1: $\mathbf{p} \leftarrow D_{\mathbb{Z}^m,\Sigma}$ with covariance matrix $\Sigma = \sigma_x^2 \cdot \mathbf{I} - \sigma_z^2 \cdot \mathbf{R} \cdot \mathbf{R}^T$
2: $\mathbf{v} = \mathbf{u} - \mathbf{A} \cdot \mathbf{p} \bmod q$.
3: $\mathbf{z} \leftarrow D_{\mathbb{Z}^\omega,\sigma_z}$ satisfying $\mathbf{G} \cdot \mathbf{z} = \mathbf{v} \bmod q$.
4: Output $\mathbf{x} = \mathbf{p} + \mathbf{R} \cdot \mathbf{z} \bmod q$.
```

to product of σ_z and $s(\mathbf{R})$, so that Σ_p is positive definite. To ensure that σ_z is set as $4 \cdot \max_i(p_i) = 4 \cdot 181 = 724$ and by **Lemma 1**, $s(\mathbf{R})$ is bounded by $4 \cdot (\sqrt{\bar{m}} + \sqrt{\omega} + 6)$ without 2^{-36} error probability in [HHSS17]. It implies that

$$\sigma_x > \sigma_z \cdot s(\mathbf{R}) = 2896 \cdot (\sqrt{\bar{m}} + \sqrt{\omega} + 6). \tag{4}$$

We state a useful lemma to estimate the size of trapdoor sampling matrices.

Lemma 1 ([MP12], Lemma 2.9)**.** *Let $\mathbf{M} \in \mathbb{R}^{n \times m}$ be a γ-sub-Gaussian random matrix with parameter σ and $s(\mathbf{M})$ be the largest singular value of $\mathbf{M}$. Then, we have $s(\mathbf{M}) \leq C \cdot \sigma \cdot (\sqrt{m} + \sqrt{n} + t)$ for some universal constant $C > 0$ except with probability at most $2\exp(\gamma)\exp(-\pi t^2)$.*