On the Invalidity of LV16/Lin17 Obfuscation Schemes: Revisited

Yupu Hu[1✉], Siyue Dong[1], Baocang Wang[1], and Xingting Dong[2]

[1] State Key Laboratory of Integrated Services Networks,
Xidian University, Xi'an, Shaanxi, China
yphu@mail.xidian.edu.cn
[2] School of Computer Science and Information Security, Guilin University Of Electronic Technology,
Guilin, Guangxi, China

Abstract. LV16/Lin17 IO schemes are famous progresses towards simplifying obfuscation mechanism. In fact, these two schemes only constructed two compact functional encryption (CFE) algorithms, while other things were taken to the AJ15 IO frame or BV15 IO frame. CFE algorithms are inserted into the AJ15 IO frame or BV15 IO frame to form a complete IO scheme. We stated the invalidity of LV16/Lin17 IO schemes. More detailedly, under reasonable assumption "real white box (RWB)" LV16/Lin17 CFE algorithms being inserted into AJ15 IO frame are insecure.

In this paper, we continue to state the invalidity of LV16/Lin17 IO schemes. The conclusion of this paper is that LV16/Lin17 CFE algorithms being inserted into BV15 IO frame are insecure. The reasoning of this paper is composed of the following three steps. First, when LV16/Lin17 CFE algorithms are inserted into secret constants. Second, when all secret random numbers are changed into the BV15 IO frame, all secret random numbers must be changed into secret constants, component functions in LV16/Lin17 CFE algorithms are cryptologic weak functions, and shapes of these component functions can be easily obtained by chosen values of independent variables. Finally, the shapes of these component functions include parameters of original function, therefore the IO scheme is insecure.

Keywords: Indistinguishability obfuscation · Multilinear maps · Yao's garbling · Randomized encoding.

1 Introduction

Indistinguishability obfuscation (IO) [1–33] makes the function unintelligent which, for arbitrarily chosen values of independent variable, provides nothing except corresponding values of the function. In the basic scene of IO there are two sides, encoding-side (also called obfuscator) and decoding-side (also called computer or client). Encoding-side presents unintelligent form $\overline{c}$ of the function c, while decoding-side chooses the value x and computes $\overline{c}(x)(= c(x))$. By

the technical limitation people don't consider "completely unintelligent" but rather "unintelligent within the group", that is, consider a public function group $\{C(\cdot,k), k \in K\}$, where k is the parameter of the group. For some secret k, encoding-side presents unintelligent form $\overline{C}(\cdot,k)$ of $C(\cdot,k)$, while decoding-side arbitrarily chooses x and computes $\overline{C}(x,k)(=C(x,k))$, but decoding-side cannot obtain any information of k by such choice and computation.

LV16/Lin17 IO schemes [19, 21] are famous progresses towards simplifying obfuscation mechanism. Notice that these two schemes did not present a complete IO scheme, but only constructed two compact functional encryption (CFE) algorithms, while other things were taken to several existing IO frames. That is, CFE algorithms are inserted into an IO frame to form a complete IO scheme. Two IO frames addressed in LV16/Lin17 IO schemes [19, 21] are AJ15 IO frame [10] and BV15 IO frame [12, 23]. The basic structures of two CFE algorithms can be described in the following way. The polynomial-time-computable Boolean function is transformed into a group of low-degree low-locality component functions by using garbling and randomized encoding, while some public combination of values of component functions is the value of the original Boolean function. The encryptor uses constant-degree multilinear maps to encrypt independent variables of component functions. The decryptor uses zero-testing tool of multilinear maps to obtain values of component functions, and then uses public combination to obtain the value of original Boolean function. We [34] stated the invalidity of the LV16/Lin17 IO schemes. More detailedly, under reasonable assumption "real white box (RWB)" LV16/Lin17 CFE algorithms being inserted into AJ15 IO frame are insecure.

In this paper we continue to state the invalidity of LV16/Lin17 IO schemes. We consider LV16/Lin17 CFE algorithms being inserted into another IO frame, that is BV15 IO frame. The conclusion of this paper is that such IO scheme is insecure. The reasoning of this paper is composed of the following three steps.

The first step, when LV16/Lin17 CFE algorithms are inserted into BV15 IO frame, all secret random numbers must be changed into secret constants. To explain this restriction, first we prove that all secret random numbers must be functions of independent variables. Second we point that, because of the limitation of existing multilinear maps [35, 36], these functions can only be constants.

The second step, when all secret random numbers are changed into secret constants, component functions in LV16/Lin17 CFE algorithms are cryptologic weak functions, and shapes of these component functions can be easily obtained by chosen values of independent variables.

The final step, shapes of these component functions include parameters of the original Boolean function, therefore the IO scheme is insecure. It does not need to combine shapes of component functions into some shape of the original Boolean function, and only shapes of component functions are enough to leak original function. The essential reason of our conclusion is that the ability of garbling and randomized encoding for protecting the original function is dependent on the ever changing random numbers, which we have clearly stated [34].

2 Preliminaries: IO, FPFE

2.1 Definition and Three Notes of Indistinguishability Obfuscation (IO)

Definition 2.1 *A uniform PPT machine IO is called an indistinguishability obfuscator [3] for a circuit class $\{C_\lambda\}$ if the following two conditions are satisfied:*

(1) Correctness. *For all security parameters λ, for all circuits $c \in C_\lambda$, for all inputs x,*$\Pr\left[\overline{c}\left(x\right) = c\left(x\right) : \overline{c} \leftarrow IO\left(\lambda, c\right)\right] = 1$

(2) Indistinguishability. *For any PPT distinguisher D, there exists a negligible function α such that the following holds. For all security parameters λ, for all pairs $c_0, c_1 \in C_\lambda$, we have that if $c_0(x) = c_1(x)$ for all inputs x, then $\left|\Pr\left[D\left(IO\left(\lambda, c_0\right)\right) = 1\right] - \Pr\left[D\left(IO\left(\lambda, c_1\right)\right) = 1\right]\right| \leq \alpha\left(\lambda\right)$.*

Note 1. The circuit class $\{C_\lambda\}$ itself should be "black box access unlearnable" (That is, the circuit class $\{C_\lambda\}$ is not cryptologic weak function class, otherwise IO is meaningless).

Note 2. $\overline{c}$ should be reusable. That is, once $\overline{c}$ is constructed, it should be fixed and repeatedly used for computing $\overline{c}(x)\,(= c(x))$ of different values of x. If $\overline{c}$ is used only once, it is much easier to be constructed, and is the component of garbling, a weaker primitive.

Note 3. The above two notes show that $\overline{c}$ is a virtual black box (VBB).

2.2 Function-private Functional Encryption (FPFE)

Functional encryption (FE) can be simply described as that the encryptor encrypts the plaintext to obtain the ciphertext, while the decryptor decrypts the ciphertext to obtain only the value of some function of the plaintext (nothing else about the information of the plaintext). Function-private FE (FPFE) is a special FE, for which the decryptor can only obtain the value of the function of the plaintext, neither other information about the plaintext nor that about the shape of the function. Brakerski [37] presented a scheme to transform an ordinary FE into an FPFE, described as the following. Suppose the function is f.

The system first takes an additional operation to hide the shape of f. It takes a secret-key encryption system $(\mathsf{SKE.Enc}, \mathsf{SKE.Dec})$, and computes $c = \mathsf{SKE.Enc}(k, f)$, $c' = \mathsf{SKE.Enc}(k', f')$, where k and k' are keys of the secret-key encryption system, f is original function, f' is another function (that is, f and f' are taken as two bit-strings, encrypted by keys k and k' respectively, and obtains c and c' respectively).

The encryptor makes use of the encryption algorithm of ordinary FE, to encrypt extended plaintext (m, m', k, k') rather than original plaintext m.

The system generates functional decryption key for the decryptor, which is of the extended function $U_{c,c'}$, rather than original function f, where $U_{c,c'}$ is

shown in Table 1. For understanding Table 1 we present following explanation: when f (as a bit-string) is known, m (as another bit-string) is known, then $f(m)$ can be obtained by a public algorithm. In other words, f (as a function) belongs to a public function group.

The decryptor makes use of the functional decryption key of the extended function $U_{c,c'}$ and the decryption algorithm of ordinary FE, to obtain $U_{c,c'}(m, m', k, k')$ $(= f(m)$, as long as $k \neq \perp)$.

Table 1. Function $U_{c,c'}$.

$U_{c,c'}(m, m', k, k')$
1.If $k \neq \perp$, compute $f \leftarrow \mathsf{SKE.Dec}(k, c)$ and output $f(m)$.
2.Else, if $k' \neq \perp$, compute $f' \leftarrow \mathsf{SKE.Dec}(k', c')$ and output $f'(m')$.
3.Else, output $\perp$.

By standard symbols, the relation between function-private functional encryption scheme $(\mathsf{FPFE.Enc}, \mathsf{FPFE.Dec})$ and ordinary functional encryption scheme $(\mathsf{FE.Enc}, \mathsf{FE.Dec})$ is described as the follow.

$$\begin{aligned} \mathsf{FPFE.Enc}(m) &= \mathsf{FE.Enc}(m, m', k, k'), \\ \mathsf{FPFE.Dec}_f &= \mathsf{FE.Dec}_{U_{c,c'}} \quad (k \neq \perp) \end{aligned}$$

3 BV15 IO frame [12, 23]

The idea of the BV15 IO frame is making use of iteration which gradually constructs IO by FE. A special point is that such FE is FPFE.

3.1 Obfuscator of the BV15 IO frame: the first step

Suppose the original Boolean function is $f(x_1, x_2, \cdots, x_n)$. The first step of the obfuscator is shown as the following equation.

$$\begin{aligned} &IO\left(f(x_1, \cdots, x_n)\right) := \mathsf{FPFE.Dec}_f, \\ &IO\left(\mathsf{FPFE.Enc}(x_1, \cdots, x_{n-1}, 0) \circ \mathsf{FPFE.Enc}(x_1, \cdots, x_{n-1}, 1)\right). \end{aligned}$$

That is, the obfuscator of n-dimensional function f is the obfuscator of an $n-1$-dimensional function and an FPFE decryption key.

Note 1 It is required that

$$\begin{aligned} &\mathsf{FPFE.Enc}(x_1, \cdots, x_{n-1}, 0) \circ \mathsf{FPFE.Enc}(x_1, \cdots, x_{n-1}, 1) \\ \neq &\{\mathsf{FPFE.Enc}(x_1, \cdots, x_{n-1}, 0), \mathsf{FPFE.Enc}(x_1, \cdots, x_{n-1}, 1)\}, \end{aligned}$$

otherwise there is no real dimension reduction.

Note 2 For the purpose of convenient description of the latter iteration process, $\mathsf{FPFE.Enc}(x_1, \cdots, x_{n-1}, 0) \circ \mathsf{FPFE.Enc}(x_1, \cdots, x_{n-1}, 1)$ is denoted as $f_{n-1}(x_1, \cdots, x_{n-1})$.

3.2 Obfuscator of the BV15 IO frame: iteration process

Suppose the present structure is

$$IO\left(f(x_1,\cdots,x_n)\right) := \mathsf{FPFE.Dec}_f, \mathsf{FPFE.Dec}_{f_{n-1}},\cdots,\mathsf{FPFE.Dec}_{f_{i+1}}, \\ IO(\mathsf{FPFE.Enc}(x_1,\cdots,x_i,0\cdots0)\circ\cdots\circ\mathsf{FPFE.Enc}(x_1,\cdots,x_i,1\cdots1)).$$

Then the next structure is

$$IO\left(f(x_1,\cdots,x_n)\right) := \mathsf{FPFE.Dec}_f, \mathsf{FPFE.Dec}_{f_{n-1}},\cdots,\mathsf{FPFE.Dec}_{f_i}, \\ IO(\mathsf{FPFE.Enc}(x_1,\cdots,x_{i-1},0\cdots0)\circ\cdots\circ\mathsf{FPFE.Enc}(x_1,\cdots,x_{i-1},1\cdots1)),$$

where $f_i(x_1,\cdots,x_i) = \mathsf{FPFE.Enc}(x_1,\cdots,x_i,0\cdots0)\circ\cdots\circ\mathsf{FPFE.Enc}(x_1,\cdots,x_i,1\cdots1)$. It is required that

$$\mathsf{FPFE.Enc}(x_1,\cdots,x_{i-1},0\cdots0)\circ\cdots\circ\mathsf{FPFE.Enc}(x_1,\cdots,x_{i-1},1\cdots1) \\ \neq \{\mathsf{FPFE.Enc}(x_1,\cdots,x_{i-1},0\cdots0),\cdots,\mathsf{FPFE.Enc}(x_1,\cdots,x_{i-1},1\cdots1),$$

otherwise there is no real dimension reduction.

3.3 Obfuscator of the BV15 IO frame: the final structure and feasibility

According to the iteration process, the final structure seems to be

$$IO\left(f(x_1,\cdots,x_n)\right) := \mathsf{FPFE.Dec}_f, \mathsf{FPFE.Dec}_{f_{n-1}},\cdots,\mathsf{FPFE.Dec}_{f_1}, \\ IO(\mathsf{FPFE.Enc}(0\cdots0)\circ\cdots\circ\mathsf{FPFE.Enc}(1\cdots1)).$$

But the hope of designers is to eliminate the symbol "IO" in the right side of the above equation, that is,

$$IO\left(f(x_1,\cdots,x_n)\right) := \mathsf{FPFE.Dec}_f, \mathsf{FPFE.Dec}_{f_{n-1}},\cdots,\mathsf{FPFE.Dec}_{f_1}, \mathsf{FPFE.Enc} \\ (0\cdots0)\circ\cdots\circ\mathsf{FPFE.Enc}(1\cdots1).$$

Only by this shape can the frame be called "IO constructed by FE". We know that if

$$\mathsf{FPFE.Enc}(0\cdots0)\circ\cdots\circ\mathsf{FPFE.Enc}(1\cdots1) \\ = \{\mathsf{FPFE.Enc}(0\cdots0),\cdots,\mathsf{FPFE.Enc}(1\cdots1)\},$$

the symbol "IO" in the right side of the above equation can be eliminated, because exhaustion itself is an IO. But another requirement is that the two sides of the above equation are not equal, otherwise the obfuscator has a set including 2^n entries, so that it is invalid. In other words, the BV15 frame has a strong requirement for "$\mathsf{FPFE.Enc}(0\cdots0)\circ\cdots\circ\mathsf{FPFE.Enc}(1\cdots1)$" structure, demanding it has the ability of exhaustion and is not an exhaustion.

Multilinear-map (that is graded encoding) makes the "$\mathsf{FPFE.Enc}(0\cdots0)\circ\cdots\circ\mathsf{FPFE.Enc}(1\cdots1)$" structure feasible. Graded encoding encodes each bit of the independent variable, and there is respectively one code for value 0 and value 1 of such bit. Therefore, $2n$ codes (rather than 2^n codes) are enough to encode 2^n values of the independent variable.

3.4 Alternative expressions of the final structure

When decoding-side arbitrarily chooses a value $(x_1, \cdots, x_n)$ of the independent variable, he "picks up" the corresponding $\mathsf{FPFE.Enc}(x_1, \cdots, x_n)$ from the "$\mathsf{FPFE.Enc}(0 \cdots 0) \circ \cdots \circ \mathsf{FPFE.Enc}(1 \cdots 1)$" structure. Then he makes use of the decryption key $\mathsf{FPFE.Dec}_{f_1}$ to decrypt $\mathsf{FPFE.Enc}(x_1, \cdots, x_n)$, and obtains a value which is still an $\mathsf{FPFE.Enc}(x_1, \cdots, x_n)$, but such $\mathsf{FPFE.Enc}$ has a different parameter, so we express it as $\mathsf{FPFE.Enc}^{(1)}(x_1, \cdots, x_n)$. In other words, we express the combination structure "$\mathsf{FPFE.Dec}_{f_1}, \mathsf{FPFE.Enc}(0 \cdots 0) \circ \cdots \circ \mathsf{FPFE.Enc}(1 \cdots 1)$" as "$\mathsf{FPFE.Enc}^{(1)}(0 \cdots 0) \circ \cdots \circ \mathsf{FPFE.Enc}^{(1)}(1 \cdots 1)$".

The above expression can be extended, and suppose i satisfies $1 \leq i \leq n-1$. When decoding-side arbitrarily chooses a value $(x_1, \cdots, x_n)$ of the independent variable, he "picks up" the corresponding $\mathsf{FPFE.Enc}(x_1, \cdots, x_n)$ from the "$\mathsf{FPFE.Enc}(0 \cdots 0) \circ \cdots \circ \mathsf{FPFE.Enc}(1 \cdots 1)$" structure. Then he sequentially makes use of the decryption keys $\mathsf{FPFE.Dec}_{f_1}, \mathsf{FPFE.Dec}_{f_2}, \cdots, \mathsf{FPFE.Dec}_{f_i}$, to repeatedly decrypt $\mathsf{FPFE.Enc}(x_1, \cdots, x_n)$, and finally obtains a value which is still an $\mathsf{FPFE.Enc}(x_1, \cdots, x_n)$. But such $\mathsf{FPFE.Enc}$ has a new parameter, so we express it as $\mathsf{FPFE.Enc}^{(i)}(x_1, \cdots, x_n)$. In other words, we express the combination structure "$\mathsf{FPFE.Dec}_{f_i}, \mathsf{FPFE.Dec}_{f_{i-1}}, \cdots, \mathsf{FPFE.Dec}_{f_1}$, $\mathsf{FPFE.Enc}(0 \cdots 0) \circ \cdots \circ \mathsf{FPFE.Enc}(1 \cdots 1)$" as "$\mathsf{FPFE.Enc}^{(i)}(0 \cdots 0) \circ \cdots \circ \mathsf{FPFE.Enc}^{(i)}(1 \cdots 1)$".

From all of the above, the final structure of the BV15 IO frame has multiple expressions, as shown in the following.

$$
\begin{aligned}
IO\left(f(x_1, \cdots, x_n)\right) :=& \mathsf{FPFE.Dec}_f, \mathsf{FPFE.Dec}_{f_{n-1}}, \cdots, \mathsf{FPFE.Dec}_{f_1}, \\
& \mathsf{FPFE.Enc}(0 \cdots 0) \circ \cdots \circ \mathsf{FPFE.Enc}(1 \cdots 1) \\
=& \mathsf{FPFE.Dec}_f, \mathsf{FPFE.Dec}_{f_{n-1}}, \cdots, \mathsf{FPFE.Dec}_{f_2}, \\
& \mathsf{FPFE.Enc}^{(1)}(0 \cdots 0) \circ \cdots \circ \mathsf{FPFE.Enc}^{(1)}(1 \cdots 1) \\
& \vdots \\
=& \mathsf{FPFE.Dec}_f, \mathsf{FPFE.Dec}_{f_{n-1}}, \cdots, \mathsf{FPFE.Dec}_{f_{i+1}}, \\
& \mathsf{FPFE.Enc}^{(i)}(0 \cdots 0) \circ \cdots \circ \mathsf{FPFE.Enc}^{(i)}(1 \cdots 1) \\
& \vdots \\
=& \mathsf{FPFE.Dec}_f, \mathsf{FPFE.Enc}^{(n-1)}(0 \cdots 0) \circ \cdots \circ \mathsf{FPFE.Enc}^{(n-1)}(1 \cdots 1)
\end{aligned}
$$

It may be said that, for different i, operations implied by the symbol "$\circ$" in the combination structure "$\mathsf{FPFE.Enc}^{(i)}(0 \cdots 0) \circ \cdots \circ \mathsf{FPFE.Enc}^{(i)}(1 \cdots 1)$" are different. The larger the i, the more complicated the operations implied by the symbol "$\circ$". For this, our answer is as follows. The BV15 IO frame itself did not clearly describe the operations implied by the symbol "$\circ$", but only implicitly required that the corresponding structure "has the ability of exhaustion and is not an exhaustion". It is easy to see that as long as the original structure "$\mathsf{FPFE.Enc}(0 \cdots 0) \circ \cdots \circ \mathsf{FPFE.Enc}(1 \cdots 1)$" satisfies such requirement, all combination structures "$\mathsf{FPFE.Enc}^{(i)}(0 \cdots 0) \circ \cdots \circ \mathsf{FPFE.Enc}^{(i)}(1 \cdots 1)$",

$i = 1, \cdots, n-1$, satisfy such requirement. This is enough for the cryptanalysis of this paper.

4 LV16/Lin17 CFE Algorithms [19, 21]

Let c be a polynomial-time computable Boolean function. Take the plaintext x.

4.1 LV16 CFE Algorithm

Operations of the encryptor (Encryption)

Step 1 (garbling)Use Yao's garbling of c [38,41,42,47–50] to construct I Boolean functions $c_i^* (x,k) = Yao_i (x, PRF(k))$, $i \in \{1, 2, \cdots, I\}$, where k is randomly chosen, PRF is a pseudorandom function.

Step 2 (randomized encoding)For each $i \in \{1, 2, \cdots, I\}$, use AIK randomized encoding of c_i^* [41–44] to construct J Boolean functions $c_{ij}^{**} (x,k,s) = AIK_{ij} (x, k, PRG(s))$, $j \in \{1, 2, \cdots, J\}$, where s is randomly chosen, PRG is a low-degree low-locality pseudorandom generator. Notice that AIK_{ij} is a low-degree low-locality Boolean function, therefore $c_{ij}^{**} (x,k,s)$ is a low-degree low-locality Boolean function.

Step 3 For each $i \in \{1, 2, \cdots, I\}$, $j \in \{1, 2, \cdots, J\}$, define function c_{ij}^{***} as

$$c_{ij}^{***} (x,k,s,b) = \begin{cases} c_{ij}^{**} (x,k,s), & for\ b = 0 \\ any\ function, & for\ b = 1 \end{cases}$$

The purpose of constructing such c_{ij}^{***} is to make so called "decryption key" complicated enough, so as to hide the shape of c_{ij}^{**}.

Step 4 (graded encoding) Up to now, each c_{ij}^{***} is a low-degree low-locality Boolean function.By using graded encoding, encode x into $\overline{x}$, and parameters (k, s, b) into $(\overline{k}, \overline{s}, \overline{b})$ (A reminder for readers: graded encoding is bitwise encoding). Submit $\{\overline{x}, \overline{k}, \overline{s}, \overline{b}\}$.

Operations of the system (Constructing functional decryption key)

By using graded encoding, encode each c_{ij}^{***} into $\overline{c_{ij}^{***}}$. Construct decoding tool (zero-testing tool) T, to guarantee $T(\bar{x}, \bar{k}, \bar{s}, \bar{b}, \overline{c_{ij}^{***}}) = c_{ij}^{***}(x,k,s,b)$. Submit $\left\{\{\overline{c_{ij}^{***}}\}, T\right\}$.

Operations of the decryptor (Decryption)

Step 1 (graded decoding) By obtained $\{\overline{x}, \overline{k}, \overline{s}, \overline{b}, \overline{c_{ij}^{***}}, T\}$, compute $T(\overline{x}, \overline{k}, \overline{s}, \overline{b}, \overline{c_{ij}^{***}}) = c_{ij}^{***}(x,k,s,b) = c_{ij}^{**}(x,k,s)$.

Step 2 (randomized decoding) Use $\{c_{ij}^{**}(x,k,s), i = 1, \cdots, I, j = 1, \cdots, J\}$ to compute $\{c_i^*(x,k), i = 1, \cdots, I\}$.

Step 3 (degarbling)Use $\{c_i^*(x,k), i = 1, \cdots, I\}$ to compute $c(x)$.

4.2 Lin17 CFE Algorithm

Operations of the encryptor (Encryption)

Step 1∼3 Same as Step 1∼3 of operations of the encryptor of LV16 scheme (see subsection 4.1).

Step 4 (increasing the number of variables to decrease the degree) Take $x \times x = \{x_u x_v\}$, and we know $x_u \cdot x_u = x_u$. Similarly, take $x \times k = \{x_u k_v\}$, $x \times s = \{x_u s_v\}$, $x \times b = \{x_u b\}$, $k \times k = \{k_u k_v\}$, $k \times s = \{k_u s_v\}$, $k \times b = \{k_u b\}$, $s \times s = \{s_u s_v\}$, $s \times b = \{s_u b\}$. For each $i \in \{1, \cdots, I\}$, $j \in \{1, \cdots, J\}$, express c_{ij}^{***} as the function of $(x \times x, x \times k, x \times s, x \times b, k \times k, k \times s, k \times b, s \times s, s \times b, b)$, rather than only the function of (x, k, s, b). That is, take an expression $c_{ij}^{****}(x \times x, x \times k, x \times s, x \times b, k \times k, k \times s, k \times b, s \times s, s \times b, b) = c_{ij}^{***}(x, k, s, b)$, then c_{ij}^{****} has a lower degree than c_{ij}^{***}.

Step 5 (graded encoding) By using graded encoding, encode $(x \times x, x \times k, x \times s, x \times b, k \times k, k \times s, k \times b, s \times s, s \times b, b)$ into $(\overline{x \times x}, \overline{x \times k}, \overline{x \times s}, \overline{x \times b}, \overline{k \times k}, \overline{k \times s}, \overline{k \times b}, \overline{s \times s}, \overline{s \times b}, \overline{b})$ (A reminder for readers: graded encoding is bitwise encoding). Submit $\{\overline{x \times x}, \overline{x \times k}, \overline{x \times s}, \overline{x \times b}, \overline{k \times k}, \overline{k \times s}, \overline{k \times b}, \overline{s \times s}, \overline{s \times b}, \overline{b}\}$.

Operations of the system (Constructing functional decryption key)

By using graded encoding, encode each c_{ij}^{****} into $\overline{c_{ij}^{****}}$. Construct decoding tool (zero-testing tool) T, to guarantee $T(\overline{x \times x}, \overline{x \times k}, \overline{x \times s}, \overline{x \times b}, \overline{k \times k}, \overline{k \times s}, \overline{k \times b}, \overline{s \times s}, \overline{s \times b}, \overline{b}, \overline{c_{ij}^{****}}) = c_{ij}^{****}(x \times x, x \times k, x \times s, x \times b, k \times k, k \times s, k \times b, s \times s, s \times b, b)$ $\big(= c_{ij}^{***}(x, k, s, b)\big)$. Submit $\big\{\{\overline{c_{ij}^{****}}\}, T\big\}$.

Operations of the decryptor (Decryption)

Step 1 (graded decoding) By obtained $\{\overline{x \times x}, \overline{x \times k}, \overline{x \times s}, \overline{x \times b}, \overline{k \times k}, \overline{k \times s}, \overline{k \times b}, \overline{s \times s}, \overline{s \times b}, \overline{b}, \{\overline{c_{ij}^{****}}\}, T\}$, compute $T(\overline{x \times x}, \overline{x \times k}, \overline{x \times s}, \overline{x \times b}, \overline{k \times k}, \overline{k \times s}, \overline{k \times b}, \overline{s \times s}, \overline{s \times b}, \overline{b}, \overline{c_{ij}^{****}}) = c_{ij}^{****}(x \times x, x \times k, x \times s, x \times b, k \times k, k \times s, k \times b, s \times s, s \times b, b) = c_{ij}^{***}(x, k, s, b)$.

Step 2∼3 Same as Step 2∼3 of operations of the decryptor of LV16 scheme (see subsection 4.1).

5 Cryptanalysis of the LV16/Lin17 CFE Algorithms Inserted into the BV15 IO Frame

5.1 Limitation of the BV15 IO frame

BV15 IO frame [12,23] did not clearly described the original structure "$\mathsf{FPFE.Enc}(0 \cdots 0) \circ \cdots \circ \mathsf{FPFE.Enc}(1 \cdots 1)$". But it did clearly show that decoding-side can arbitrarily choose value $(x_1, \cdots, x_n)$ of the independent variable and "pick up" a corresponding value $\{\mathsf{FPFE.Enc}(x_1, \cdots, x_n), \mathsf{FPFE.Enc}^{(1)}(x_1, \cdots, x_n), \cdots, \mathsf{FPFE.Enc}^{(n-1)}(x_1, \cdots, x_n)\}$ from the original and combination structures. That is, when decoding-side repeatedly chooses a value $(x_1, \cdots, x_n)$ of the independent variable, "picked up" values of $\{\mathsf{FPFE.Enc}(x_1, \cdots, x_n), \mathsf{FPFE.Enc}^{(1)}(x_1, \cdots, x_n), \cdots, \mathsf{FPFE.Enc}^{(n-1)}(x_1, \cdots, x_n)\}$ are equal. This directly derives the following Proposition 5.1.

Proposition 5.1 *For the BV15 IO frame, all secret random numbers used by* $\{\mathsf{FPFE.Enc}(x_1,\cdots,x_n), \mathsf{FPFE.Enc}^{(1)}(x_1,\cdots,x_n),\cdots, \mathsf{FPFE.Enc}^{(n-1)}(x_1,\cdots,x_n)\}$ *should be secret functions of* $(x_1,\cdots,x_n)$*. The so-called "random numbers" include all variables rather than just* $(x_1,\cdots,x_n)$*.*

5.2 Limitation of graded encoding

Suppose the FE algorithms inserted into the BV15 IO frame are constructed by graded encoding (just like the LV16/Lin17 CFE algorithms [19,21]). Suppose r is a bit of random numbers of $\mathsf{FE.Enc}$. For the purpose of r satisfying Proposition 5.1, the encoding-side presents N codes $\{\overline{r_1},\cdots,\overline{r_N}\}$ of r. A part of these N codes are codes of $r=0$, and another part are codes of $r=1$, while decoding-side does not know the corresponding value of r for each code. When decoding-side chooses a value $(x_1,\cdots,x_n)$ of the independent variable, he is permitted to use only one code $\overline{r_i}$ from $\{\overline{r_1},\cdots,\overline{r_N}\}$, and prohibited to use other codes. That is, although decoding-side does not know the value of r corresponding to the code $\overline{r_i}$, he does know the value of r satisfies Proposition 5.1. In other words, such value of r is a function value of such $(x_1,\cdots,x_n)$, while the function is secretly defined by the encoding-side.

This design idea immediately gives rise to an unsolvable problem: the encoding-side cannot prevent decoding-side from abusing codes. When decoding-side chooses $(x_1,\cdots,x_n)$, where originally $\overline{r_i}$ is a permitted code and $\overline{r_j}$ is a prohibited code, the encoding-side lacks a method to prevent decoding-side to use $\overline{r_j}$ rather than $\overline{r_i}$. Why? Because existing graded encoding schemes [35, 36] failed to provide a prohibition barrier for the value of the independent variable. More detailedly, existing graded encoding schemes [35,36] can only provide a prohibition barrier for the input value of a concrete gate.

For the purpose of avoiding the security weakness caused by such abuse, we can only set $N=1$, that is r has only one code $\bar{r}$. Therefore, the following Proposition 5.2 holds.

Proposition 5.2 *Suppose the FE schemes inserted into the BV15 IO frame are constructed by graded encoding. Then all secret random numbers used by* $\{\mathsf{FPFE.Enc}(x_1,\cdots,x_n), \mathsf{FPFE.Enc}^{(1)}(x_1,\cdots,x_n),\cdots, \mathsf{FPFE.Enc}^{(n-1)}(x_1,\cdots,x_n)\}$ *should be secret constants. The so-called "random numbers" include all variables rather than just* $(x_1,\cdots,x_n)$*.*

5.3 Cryptanalysis of the LV16/Lin17 CFE algorithms inserted into the BV15 IO frame

Suppose the FE algorithms inserted into the BV15 IO frame are the LV16/Lin17 CFE algorithms. Consider the frame expressed by the last composition structure:

$$IO\left(f(x_1,\cdots,x_n)\right) := \mathsf{FPFE.Dec}_f, \mathsf{FPFE.Enc}^{(n-1)}(0\cdots 0)\circ\cdots\circ \mathsf{FPFE.Enc}^{(n-1)}(1\cdots 1).$$

According to the statement of subsection 2.2, decoding-side knows the shape of the extended function $U_{c,c'}$, so he knows the value of c. To hide f from

decoding-side, it is necessary to hide k from him (Because $c = \mathsf{SKE.Enc}(k, f), f = \mathsf{SKE.Dec}(k, c)$).

On the other hand, all variables other than $(x_1, \cdots, x_n)$ in $\mathsf{FPFE.Enc}^{(n-1)}(x_1, \cdots, x_n)$ should be constant (See Proposition 5.2). Therefore, to hide k from decoding-side, total $\mathsf{FPFE.Dec}$ process cannot appear "cryptologic weak function". However, according to the LV16/Lin17 CFE algorithms, decoding-side firstly obtains the values of component functions. Component functions are all "cryptologic weak functions", and their shapes can be obtained by choosing values of $(x_1, \cdots, x_n)$. Shapes of component functions include information of k, so k cannot be hidden from decoding-side.

Conclusion: The LV16/Lin17 CFE algorithms inserted into the BV15 IO frame is insecure.

Additional comment: the ability of garbling and randomized encoding for protecting original function is dependent on the ever changing random numbers, which we have clearly stated [34].

References

1. Barak, B., Goldreich, O., Impagliazzo, R., Rudich, S., Sahai, A., Vadhan, S., Yang, K.: On the (im)possibility of obfuscating programs. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 1-18. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44647-8_1
2. Lynn, B., Prabhakaran, M., Sahai, A. (2004).: Positive results and techniques for obfuscation. In Advances in Cryptology-EUROCRYPT 2004: International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004. Proceedings 23 (pp. 20-39). Springer Berlin Heidelberg.
3. Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: 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, pp. 40-49 (2013).
4. Brakerski, Z., Rothblum, G.N.: Virtual black-box obfuscation for all circuits via generic graded encoding. In: Lindell, Y. (ed.) TCC 2014. LNCS, vol. 8349, pp. 1-25. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-54242-8_15
5. Barak, B., Garg, S., Kalai, Y.T., Paneth, O., Sahai, A.: Protecting obfuscation against algebraic attacks. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 221-238. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-55220-5_13
6. Pass, R., Seth, K., Telang, S.: Indistinguishability obfuscation from semantically secure multilinear encodings. In: Gary, J.A., Gennaro, R. (eds.) CRYPTO 2014. LNCS, vol. 8616, pp. 500-517. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44371-2_28
7. Ananth, P.V., Gupta, D., Ishai, Y., Sahai, A.: Optimizing obfuscation: avoiding Barrington's theorem. In: ACM CCS 2014, Scottsdale, AZ, USA, pp. 646-658, 3-7 November 2014.
8. Applebaum, B., Brakerski, Z.: Obfuscating circuits via composite-order graded encoding. In: Y. Dodis, Nielsen, J.B. (eds.) TCC 2015. LNCS, vol. 9015, pp. 528-556. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46497-7_21

9. Zimmerman, J.: How to obfuscate programs directly. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9057, pp. 439-467. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46803-6_15
10. Ananth, P., Jain, A.: Indistinguishability obfuscation from compact functional encryption. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015, Part I. LNCS, vol 9215, pp 308-326. Springer, Berlin, Heidelberg (2015). https://doi.org/10.1007/978-3-662-47989-6_15
11. Gentry, C., Lewko, A.B., Sahai, A., Waters, B.: Indistinguishability obfuscation from the multilinear subgroup elimination assumption. In: Guruswami, V. (ed.) 56th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2015, 17-20 October 2015, Berkeley, CA, USA, pp. 151-170 (2015).
12. Bitansky, N., Vaikuntanathan, V.: Indistinguishability obfuscation from functional encryption. In: Guruswami, V. (ed.) 56th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2015, 17-20 October 2015, Berkeley, CA, USA, pp. 171–190(2015).
13. Canetti, R., Kalai, Y. T., Paneth, O.: On obfuscation with random oracles. Cryptology ePrint Archive. 2015,048.
14. Mahmoody, M., Mohammed, A., Nematihaji, S.: More on Impossibility of Virtual Black-Box Obfuscation in Idealized Models. IACR Cryptol. ePrint Arch., 2015, 632.
15. Pass, R., Shelat, A.: Impossibility of VBB obfuscation with ideal constant-degree graded encodings. In Theory of Cryptography: 13th International Conference, TCC 2016-A, Tel Aviv, Israel, January 10-13, 2016, Proceedings, Part I (pp. 3-17). Berlin, Heidelberg: Springer Berlin Heidelberg.
16. Bitansky, N., Vaikuntanathan, V.: Indistinguishability obfuscation: from approximate to exact. In Theory of Cryptography: 13th International Conference, TCC 2016-A, Tel Aviv, Israel, January 10-13, 2016, Proceedings, Part I 13 (pp. 67-95). Springer Berlin Heidelberg.
17. Miles, E., Sahai, A., Zhangdry, M.: Annihilation attacks for multilinear maps: cryptanalysis of indistinguishability obfuscation over GGH13. In: IACR Cryptology ePrint Archive, vol. 2016, p. 147 (2016).
18. Lin, H.: Indistinguishability obfuscation from constant-degree graded encoding schemes. In: Fischlin, M., Coron, J.S. (eds.) EUROCRYPT 2016, LNCS, vol.9665, pp. 28–57. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49890-3_2
19. Lin, H., Vaikuntanathan, V.: Indistinguishability obfuscation from DDH-like assumptions on constant-degree graded encodings. In: 2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS) (pp. 11-20). IEEE.
20. Ananth, P., Sahai, A.: Projective arithmetic functional encryption and indistinguishability obfuscation from degree-5 multilinear maps. In: Coron, JS., Nielsen, J. (eds.) EUROCRYPT 2017, Part I. LNCS, vol 10210, pp 152-181. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-56620-7_6
21. Lin, H.: Indistinguishability obfuscation from SXDH on 5-linear maps and locality-5 PRGs. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017, Part I. LNCS, vol.10401, pp. 599-629. Springer, Heidelberg (2017). https://doi.org/10.1007/978-3-319-63688-7_20
22. Lin, H., Tessaro, S.: Indistinguishability obfuscation from trilinear maps and blockwise local PRGs. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017, Part I. LNCS, vol. 10401, pp. 630–660. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63688-7_21
23. Bitansky, N., Vaikuntanathan, V.: Indistinguishability obfuscation from functional encryption. Journal of the ACM (JACM), 65(6), 1-37.

24. Ananth, P., Jain, A., Sahai, A.: Indistinguishability obfuscation without multilinear maps: iO from LWE, bilinear maps, and weak pseudorandomness. Cryptology ePrint Archive, Report 2018/615 (2018).
25. Gentry, C., Jutla, C.S., Kane, D.: Obfuscation Using Tensor Products. In: Electronic Colloquium on Computational Complexity (ECCC), vol. 25 (2018).
26. Lin, H., Matt, C.: Pseudo Flawed-Smudging Generators and Their Application to Indistinguishability Obfuscation. Cryptology ePrint Archive, Report 2018/646 (2018).
27. Agrawal, S.: Indistinguishability obfuscation without multilinear maps: new methods for bootstrapping and instantiation. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019, Part I. LNCS, vol 11476, pp. 191-225. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-17653-2_7
28. Jain, A., Lin, H., Matt, C., Sahai, A.: How to leverage hardness of constantdegree expanding polynomials over R to build iO. In: Ishai, Y., Rijmen, V.(eds.) EUROCRYPT 2019, Part I. LNCS, vol. 11476, pp. 251-281. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-17653-2_9
29. Bartusek, J., Lepoint, T., Ma, F., Zhandry, M.: New techniques for obfuscating conjunctions. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019, Part III. LNCS, vol 11478, pp 636-666. Springer, Cham(2019). https://doi.org/10.1007/978-3-030-17659-4_22
30. Ananth, P., Jain, A., Lin, H., Matt, C., Sahai, A.: Indistinguishability obfuscation without multilinear maps: new paradigms via low degree weak pseudorandomness and security amplification. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019, Part III. LNCS, vol 11694, pp 284-332. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-26954-8_10
31. Agrawal, S., Pellet-Mary, A.: Indistinguishability obfuscation without maps: attacks and fixes for noisy linear FE. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020, Part I. LNCS, vol 12105, pp. 110-140. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45721-1_5
32. Brakerski, Z., Döttling, N., Garg, S., Malavolta, G.: Candidate iO from homomorphic encryption schemes. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020, Part I. LNCS, vol 12105, pp. 79-109. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45721-1_4
33. Bartusek, J., Ishai, Y., Jain, A., Ma, F., Sahai, A., Zhandry, M.: Affine determinant programs: A framework for obfuscation and witness encryption. In: 11th Innovations in Theoretical Computer Science Conference (ITCS 2020). Schloss Dagstuhl-Leibniz-Zentrum für Informatik, (2020).
34. Hu, Y., Siyue, D., Baocang, W., Xingting, D.: On the Invalidity of LV16/Lin17 Obfuscation Schemes. Cryptology ePrint Archive.
35. Garg, S., Gentry, C., Halevi, S.: Candidate multilinear maps from ideal lattices. In: T. Johansson, P.Q. Nguyen (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 1-17. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38348-9_1
36. Coron, J.-S., Lepoint, T., Tibouchi, M.: Practical multilinear maps over the integers. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 476–493. Springer, Heidelberg (2013)
37. Brakerski, Z., Segev, G.: Function-private functional encryption in the private-key setting. In: Journal of Cryptology, 31, 202-225.
38. Goldwasser, S., Kalai, Y.T., Popa, R.A., Vaikuntanathan, V., Zeldovich, N.: Reusable garbled circuits and succinct functional encryption. In: STOC, pp. 555–564, 2013.

39. Agrawal, S.: Stronger security for reusable garbled circuits, general definitions and attacks. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017. LNCS, vol. 10401, pp. 3-35. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63688-7_1
40. Hu, Y, P., Dong, S, Y., Wang, B, C., Liu, J.: Notes on Reusable Garbling. Cryptology ePrint Archive, Paper 2022/1208.https://eprint.iacr.org/2022/1208
41. Ishai, Y., Kushilevitz, E.: Randomizing Polynomials: A new representation with applications to round-efficient secure computation. In: Proceedings of the 41st FOCS, pp. 294-304 (2000).
42. Ishai, Y., Kushilevitz, E.: Perfect constant-round secure computation via perfect randomizing polynomials. In: Widmayer, P., Eidenbenz, S., Triguero, F., Morales, R., Conejo, R., Hennessy, M., (eds.) ICALP 2002. LNCS, vol. 2380, pp. 244-256. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45465-9_22
43. Applebaum, B., Ishai, Y., Kushilevitz, E.: Cryptography in nc0. In: Proceedings of the 45th FOCS, pp. 166-175 (2004).
44. Applebaum, B., Ishai, Y., Kushilevitz, E.: Computationally private randomizing polynomials and their applications. In: Computational Complexity, vol. 15 (2006), pp. 115 – 162. https://doi.org/10.1007/s00037-006-0211-8
45. Barrington, D.A.: Bounded-width polynomial-size branching programs recognize exactly those languages in NC1. In: Journal of computer and system sciences, vol. 38, no. 1, pp. 150 – 164 (1989), Elsevier 1989.
46. Gorbunov, S., Vaikuntanathan, V., Wee, H.: Attribute-based encryption for circuits. In: Boneh, D., Roughgarden, T., Feigenbaum, J. (eds.) 45th ACM STOC, pp. 545-554. ACM Press, June 2013.
47. Yao, A.C.: Protocols for secure computations (extended abstract). In: Proceedings of the 23th FOCS, pp. 160-164 (1982).
48. Yao, A.C.: How to generate and exchange secrets (extended abstract). In: Proceedings of the 27th FOCS, pp. 162-167 (1986).
49. Bellare, M., Hoang, V.T., Rogaway, P.: Foundations of garbled circuits. In: Proceedings of the 2012 ACM conference on Computer and Communications Security (CCS), pp. 784-796. October 2012.
50. Gentry, C., Gorbunov, S., Halevi, S., Vaikuntanathan, V., Vinayagamurthy, D.: How to compress (reusable) garbled circuits. In: IACR eprint 2013/687.