On the Invalidity of LV16/Lin17 Obfuscation Schemes

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. Indistinguishability obfuscation (IO) is at the frontier of cryptography research for several years. LV16/Lin17 obfuscation 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 AJ15 IO frame or BV15 IO frame. That is, CFE algorithms are inserted into AJ15 IO frame or BV15 IO frame to form a complete IO scheme. The basic structure 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 randomized encoding, while some public combination of values of component functions is the value of original Boolean function. The encryptor uses constant-degree multilinear maps (rather than polynomial-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 (rather than to obtain values of independent variables), and then uses public combination to obtain the value of original Boolean function.

In this paper we restrict IO to be a real white box (RWB). Under such restriction we point out that LV16/Lin17 CFE algorithms being inserted into AJ15 IO frame are invalid. More detailedly, such insertion makes the adversary gradually learn the shape of the function, therefore the scheme is not secure. In other words, such scheme is not a real IO scheme, but rather a garbling scheme. It needs to be said that RWB restriction is reasonable, which means the essential contribution of IO for cryptography research.

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

1 Introduction

Indistinguishability obfuscation (IO) 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.

IO was first defined by Barak et al [1], and has received a lot of attentions in the community [2–32]. To describe our work clearly, in this section we carefully state those items related to IO, including the restriction on the function group, virtual black box (VBB), real white box (RWB), the difference with garbling, and bootstrapping.

1.1 The restriction on the function group

For constructing an IO scheme, the public function group $\{C(\cdot, k), k \in K\}$ should not be learnable by black box access, otherwise the IO scheme is meaningless [1]. In other words, any function in the group should not leak the information of the parameter k by polynomially many chosen values of independent variable and corresponding values of the function. In still other words, the public function group should not be a cryptographic weak function group. For example, a linear function group is a cryptographic weak function group, a function group with a parameter space K too small or with an independent variable range too small is a cryptographic weak function group, and so on.

1.2 Virtual black box (VBB)

Virtual black box (VBB) means that having IO scheme access to the function doesn't obtain more information than having black box access to it. Because of the restriction in subsection 1.1 that the function group is "unlearnable by the black box access", VBB implies that "IO access and black box access are equally unlearnable". VBB property is difficult to be proven, and several papers [1,13–15] pointed function groups that cannot obtain VBB IO schemes. Even though, VBB is still the wish for constructing IO schemes, at least "seemingly VBB".

1.3 Real white box (RWB)

Real white box (RWB) means that, for decoding-side, each step of access to the function by the IO scheme is not a black box access (note that small sized input-output table should be taken as a white box rather than a black box, for example, the S-box of a block cipher). RWB property is not mentioned in the IO literature, and several papers [2,13,15] clearly made use of black box in their constructions of IO. We argue that RWB property is the essential contribution of

IO for cryptography. If the IO scheme permits black box access, the simplest IO scheme is taking the function itself as a black box, which is meaningless. What is more, the following two evidences show the reasonability of RWB property for IO schemes.

The first evidence comes from the application field of IO. Barak et al [1] and Lynn et al [2] list a challenging application "transforming private-key encryption into public-key encryption". Barak et al [1] detailedly state such application: to publish an obfuscation of the private-key encryption function to allow everyone encrypt, yet only one possessing the encryption function (that is, the owner of the key) should be able to decrypt. That is, the encryptor of such public-key cryptosystem plays the role of "decoding-side of an IO scheme" (rather than the role of "encoding-side of an IO scheme"). Here a keypoint is the security for untrusted environment, that is, the encryptor should be sure that the whole encryption procedure is his own computation rather than black box access to an unknown computation. From existing public-key encryption functions, there is no black box, the difference between authorities of the encryptor and the decryptor only depends on the trapdoor. It may be said that the encryptor of the public-key cryptosystem calls random number by black box access. Our explanation is that such a special "black box" is only used for relaxing the computation of the encryptor rather than hiding something from him. In other words, the encryptor can generate random numbers by tossing coins without accessing to such special "black box".

The second evidence comes from two complete IO schemes, GGH+13b [3] and AB15 [8]. The former is called the first IO candidate scheme [12, 16], while the latter is the unique IO scheme directly constructed by multilinear maps [33, 34]. Such two schemes satisfy RWB property.

1.4 The difference between IO and garbling

The functionality of IO and garbling are quite similar, both of which can make the function unintelligent. The unique difference between such two cryptographic primitives is that an IO scheme is a reusable scheme, while a garbling is a one-time scheme. Decoding-side of an IO scheme can repeatedly use an unintelligent form to compute corresponding values of the function for different values of independent variable, without contacting encoding-side for each computation. In a garbling scheme an unintelligent form can be used only once, otherwise the shape of the function may be leaked (There was a "reusable garbling scheme" [35, 36], but we pointed [37] that it is still a one-time scheme). This is the reason of the following three facts.

Fact 1 Garbling is much easier to be constructed than IO. The former does not need to consider VBB/RWB, nor to use novel cryptographic tools. While the latter needs not only to consider both VBB and RWB, but also to use multilinear maps [33, 34], maybe fully-homomorphic encryption (FHE), zero-knowledge proof (ZK), and so on.

Fact 2 Garbling is much more mature than IO. The former has complete security proof, while each candidate of the latter is not clear about the security.

Fact 3 Garbling has much fewer applications than IO. The former is mainly applied for multi-party computation (MPC) and some similar scenes of one-time interaction, while the latter, if exists, is a revolutionary advance in public-key cryptography (A note: the sentence "IO is a revolutionary advance in public-key cryptography" is an exaggeration. If only the IO scheme doesn't include any trapdoor of existing public-key ciphers, can it be called a revolutionary advance. The present situation is that IO schemes are inseparable from those existing trapdoors, such as LWE).

1.5 Bootstrapping

Major technical basis of IO is multilinear maps [33, 34], also called graded encoding. A simple consensus is that, when the function has its circuit size N, an IO scheme can be constructed by $O(N)$ degree graded encoding, which can be seen in AB15 [8]. Two obstacles are that high degree graded encoding has huge size, and that the security of graded encoding is hard to be proven.

The first question people try to ask is whether IO scheme can be constructed without graded encoding. After several efforts [23, 26, 29, 30] there is no clear answer.

The second question they try to ask is that can an IO scheme be constructed for the function with large circuit size by low degree graded encoding. More specifically, can it be constructed for the function with any circuit size by constant degree graded encoding. Such question is called "bootstrapping" and, to some extent, has a positive answer.

In fact, it was found that the degree of graded encoding needn't match the circuit size N of the function, but rather the circuit depth d. That is, IO scheme can be constructed by $O(d)$ degree graded encoding (needn't be constructed by $O(N)$ degree graded encoding). GGH+13b IO scheme [3] sufficiently made use of such feature, to express the P/poly function by an appropriate NC^1 circuit (simply speaking, essentially decrease the circuit depth). The technical reason is that the decryption circuit of fully-homomorphic encryption (FHE) and zero knowledge proof (ZK) can help the function essentially decrease its circuit depth, although its circuit size tends larger.

Another routine of bootstrapping is making use of functional encryption (FE) to construct IO schemes [10, 12, 18–21], which is the focus of this paper.

1.6 About LV16/Lin17 IO schemes and the work of this paper

LV16/Lin17 IO schemes [19, 21] are famous progresses towards simplifying obfuscation mechanism. These two schemes belong to the second routine of bootstrapping, making use of functional encryption (FE) to construct IO. Detailedly speaking, 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 some existing IO frames. That is, CFE algorithms are inserted into an IO frame to form a complete IO scheme. Two IO frames which were addressed in the LV16/Lin17 IO schemes [19, 21] are AJ15 IO frame [10] and BV15 IO frame [12]. 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 randomized encoding [38–41], while some public combination of values of component functions is the value of original Boolean function. The encryptor uses constant-degree multilinear maps (rather than polynomial-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 (rather than to obtain values of independent variables), and then uses public combination to obtain the value of original Boolean function.

In this paper we restrict IO to be a real white box (RWB), and consider the case where LV16/Lin17 CFE algorithms are inserted into AJ15 IO frame [10]. AJ15 frame is originally a multi-input functional encryption (MIFE) frame, which needs many CFE algorithms as components to form a complete MIFE scheme. Such MIFE frame is restated to become an IO frame, and such complete MIFE scheme is restated to become a complete IO scheme. We have following three observations.

Observation 1 In this case encoding-side/decoding-side of IO is just encryptor/decryptor of CFE. The difference is that, the secret random number used by CFE encryptor should in IO construction be either predetermined and not be changed, or only the function of related bits of independent variable, otherwise the IO scheme violates either RWB or reusability.

Observation 2 Component functions in LV16/Lin17 CFE algorithms are too simple. When secret random numbers are changed into secret fixed numbers, shapes of these component functions can be solved by a small number of values of independent variables and corresponding component functions (or maybe solving their equivalent shapes).

Observation 3 When shapes of all component functions are known, some information about the shape of original Boolean function can be obtained by the public combination. In other words, if secret random numbers in component functions are changed into secret fixed numbers, randomized encoding doesn't promise to protect the shape of original Boolean function, which can be seen from existing randomized encoding schemes IK00/IK02/AIK04/AIK06 [38–41] and Appendix A of this paper.

By the three observations above, the scheme with LV16/Lin17 CFE algorithms inserted into AJ15 IO frame is invalid. More detailedly, such scheme under RWB requirement may make the adversary gradually learn the shape of original Boolean function, so that it is not secure. It can only be used for the scene with secret numbers ever changing, so that it is a garbling scheme rather than an IO scheme. Indeed LV16/Lin17 have many novel designs, including arithmetic circuit, pseudo-random-generator (PRG), low-degree partial function, punching technique, and so on. But these novel designs are only for simplifying the computation rather than for avoiding our three observations.

Besides, this paper takes attention to a conclusion of AJ15 [10] and LV16/Lin17 [19, 21]. The conclusion is that "if CFE is secure, IO can be constructed". Such conclusion is at least not strict. A stricter conclusion should be that "if CFE encryption function is secure, IO can be constructed". Here a CFE encryption function is the CFE encryption algorithm with the secret number fixed rather than ever changing, which can be seen from the latter of this paper.

In order to avoid any misunderstanding and refute any exculpatory of the authors of the LV16/Lin17 schemes, this paper presents a large number of notes and explanations. Yet this paper does not discuss the case where the LV16/Lin17 CFE are inserted into BV15 [12], another IO frame, which will be presented in another paper.

2 Preliminaries: IO, Graded Encoding, Garbling, Randomized Encoding, and FE

2.1 Definition and Four 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 $\{C_\lambda\}$ should be "black box access unlearnable" (see the statement in subsection 1.1 of this paper).

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, see the statement in subsection 1.2 of this paper).

Note 4. $\overline{c}$ should be a real white box (RWB, see the statement in subsection 1.3 of this paper).

2.2 Graded Encoding

Graded encoding (also called multi-linear map) [33, 34] is the most important component of IO and sometimes unique component (That is, sometimes graded encoding itself is IO).

For a Boolean circuit c, operations of encoding-side are as follows. He encodes c into $\overline{c}$, and encodes independent variable range X into $\overline{X}$ (That is, encodes each value from $\{0,1\}$ of each entry X_i of $X = (X_1, \cdots, X_n)$. So that $\overline{X} = (\overline{X_{1,0}}, \overline{X_{1,1}}, \cdots, \overline{X_{n,0}}, \overline{X_{n,1}})$. He also constructs the decoding tool T (also called zero-testing tool, which can test value just at one point, neither sooner nor later). Then he submits $\{\overline{X}, \overline{c}, T\}$.

Decoding-side obtains $\{\overline{X}, \overline{c}, T\}$, and chooses $\overline{x}$ from $\overline{X}$ according to his choice of x. Then he can compute $T(\overline{x}, \overline{c})$ which is equal to $c(x)$.

The first challenge is to guarantee $T(\overline{x}, \overline{c}) = c(x)$ for any x with leaking nothing else, and the second challenge is to reduce the huge size. All candidates of graded encoding are not very sure to face such two challenges, so that an effort is to ease the role of graded encoding in the IO structure (see the statement in subsection 1.5 of this paper).

2.3 About Garbling and Randomized Encoding

Garbling [38–41, 44, 45] can be taken as a one-time version of IO, so that it is much simpler than IO. Although some garbling schemes [38] have limited ability for small number of reusability, no garbling candidate can be taken as a true reusable scheme. Besides, there was a "reusable garbling scheme" [35, 36], but we pointed [37] it is still a one-time scheme.

A technical difference of garbling is that, $\overline{X}$ is not completely sent to decoding-side, but rather by using "one-out-two oblivious transfer". That is, for each entry X_i of $X = (X_1, \cdots, X_n)$, decoding-side can and only can choose to receive one from $\{\overline{X_{i,0}}, \overline{X_{i,1}}\}$, while encoding-side doesn't know the choice of decoding-side.

Randomized encoding [38–41] is a special type of garbling, to express a function in terms of a group of low-degree low-locality functions with random parameters.

2.4 FE

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). A consensus is that FE is a simpler cryptographic primitive than IO, and easier to be proven the security, so that a research item is constructing IO by FE [10, 12, 18, 19, 21]. AJ15 [10] made use of multi-input FE (MIFE) as the frame, compact FE (CFE) as the components, function-private FE (FPFE) as the technique, to construct IO scheme.

3 Cryptanalysis of AJ15 IO frame

3.1 Why MIFE

Multi-input FE (MIFE) is such special FE, that the plaintext is separated into many parts which are respectively encrypted into ciphertext parts. For decryption, these ciphertext parts are combined to compute the value of the function of the whole plaintext. Such MIFE structure is easy to be transformed into IO structure, because decoding-side of IO scheme needs selective decoding ability rather than simple decoding ability. More detailedly, decoding-side arbitrarily chooses values of independent variable and computes corresponding values of the function.

3.2 FPFE

Function-private FE (FPFE) is another 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 [48] 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 $\perp$.

The decryption algorithm the decryptor uses is of ordinary FE, so that he can obtain the shape of the extended function $U_{c,c'}$ and the value of c. He cannot obtain k, because k is a part of the plaintext. Therefore he cannot obtain f $(f = \mathsf{SKE.Dec}(k, c))$. That means that the scheme satisfies the requirement of FPFE.

The price is huge. First, to hide f from the decryptor. f is extended into (c, k), where k is only known by the encryptor, and c only by the decryptor.

Then, to make f deeply hidden, another function f' is extended into (c', k'), where k' is only known by the encryptor, and c' only by the decryptor. Finally, the decryptor only knows (c, c') which is the parameter of much bigger function of much longer plaintext, making it much more complicated to generate functional decryption key for such function.

3.3 AJ15 MIFE frame [10]

AJ15 MIFE frame is the starting point of AJ15 IO frame [10]. This frame makes use of Brakerski FPFE transformation [48] to realize such functionality: the plaintext is separated into $N+1$ parts to be encrypted respectively, the decryptor combines $N+1$ ciphertexts to compute the value of the function of the plaintext entity, and at same time it should be guaranteed not to leak more than an ordinary FE frame does. In AJ15 [10], the description of MIFE frame is an iteration, where “arity-amplification” and “function-privacy” are implemented alternately. We point out that their description has a mistake. More detailedly, their definition of function (Fig.2 of [10]) makes the iteration unsustainable. However, this mistake is not essential, and only needs a minor modification. We restate AJ15 MIFE frame in this subsection.

This frame needs $N+1$ ordinary FE schemes, { $\mathrm{FE}^{(0)}$, $\mathrm{FE}^{(1)}$,$\cdots$, $\mathrm{FE}^{(N)}$} (For the purpose of simply stating, encryption keys of these FE schemes are omitted). This frame needs a secret-key encryption system $(\mathsf{SKE.Enc}, \mathsf{SKE.Dec})$. This frame needs a pseudo-random function PRF. This frame also gives the encryptor a special ability which is constructing functional decryption keys for $\{\mathrm{FE}^{(1)},\cdots, \mathrm{FE}^{(N)}\}$ (But cannot construct functional decryption key for $\mathrm{FE}^{(0)}$). This frame is described as follows. Suppose the function is f.

Operations of the encryptor (Encryption)

Step 1 Take the plaintext m, and separate it into $N+1$ parts: $m = (m^{(0)}, m^{(1)}, \cdots, m^{(N)})$.

Step 2 Randomly choose $(m'^{(1)}, \cdots, m'^{(N)}), ((k^{(0)}, k'^{(0)}), (k^{(1)}, k'^{(1)}), \cdots, (k^{(N-1)}, k'^{(N-1)})), (\tau^{(1)}, \cdots, \tau^{(N)}), K$.

The purpose of choosing these random numbers is to make FE schemes in the frame encrypt extended plaintexts. The extended plaintexts FE schemes encrypt are as shown in Table 2.

Table 2. Extended plaintexts to be encrypted by FE schemes

FE scheme	The extended plaintext to be encrypted
$\mathsf{FE}^{(N)}.\mathsf{Enc}$	$\Big((k^{(0)}, k'^{(0)}), \cdots, (k^{(N-1)}, k'^{(N-1)}), (m^{(N)}, m'^{(N)}, 1, \tau^{(N)}, N)\Big)$
$\mathsf{FE}^{(N\text{-}1)}.\mathsf{Enc}$	$\big((k^{(0)}, k'^{(0)}), \cdots, (k^{(N-2)}, k'^{(N-2)}), (m^{(N-1)}, m'^{(N-1)}, 1, \tau^{(N-1)}, N-1), (m^{(N)}, m'^{(N)}, 1, \tau^{(N)}, N)\big)$
$\vdots$	$\ldots$
$\mathsf{FE}^{(1)}.\mathsf{Enc}$	$\Big((k^{(0)}, k'^{(0)}), (m^{(1)}, m'^{(1)}, 1, \tau^{(1)}, 1), \cdots, (m^{(N)}, m'^{(N)}, 1, \tau^{(N)}, N)\Big)$
$\mathsf{FE}^{(0)}.\mathsf{Enc}$	$\Big((m^{(0)}, \cdots, m^{(N)}), PRF(K, \tau^{(1)}, \cdots, \tau^{(N)})\Big)$

Step 3 Compute $M^{(N)} \leftarrow \mathsf{FE}^{(N)}.\mathsf{Enc}((k^{(0)}, k'^{(0)}), \cdots, (k^{(N-1)}, k'^{(N-1)}), (m^{(N)}, m'^{(N)}, 1, \tau^{(N)}, N))$

Step 4 For $i = 1, \cdots, N-1$, define the function $f^{(i)}$ as the following.

$$\begin{aligned}
&f^{(i)}((k^{(0)}, k'^{(0)}), \cdots, (k^{(i-1)}, k'^{(i-1)}), (m^{(i+1)}, m'^{(i+1)}, 1, \tau^{(i+1)}, i+1), \cdots, \\
&(m^{(N)}, m'^{(N)}, 1, \tau^{(N)}, N)) \\
=&\mathsf{FE}^{(i)}.\mathsf{Enc}((k^{(0)}, k'^{(0)}), \cdots, (k^{(i-1)}, k'^{(i-1)}), (m^{(i)}, m'^{(i)}, 1, \tau^{(i)}, i), (m^{(i+1)}, m'^{(i+1)}, \\
&1, \tau^{(i+1)}, i+1), \cdots, (m^{(N)}, m'^{(N)}, 1, \tau^{(N)}, N))
\end{aligned}$$

(A reminder for readers: $(m^{(i)}, m'^{(i)}, 1, \tau^{(i)}, i)$ is a part of parameters of the function $f^{(i)}$, rather than a part of independent variables). Then randomly choose another function $f'^{(i)}$.

Step 5 Define the function $f^{(0)}$ as the follow.

$$\begin{aligned}
&f^{(0)}\left((m^{(1)}, m'^{(1)}, 1, \tau^{(1)}, i_1), \cdots, (m^{(N)}, m'^{(N)}, 1, \tau^{(N)}, i_N)\right) \\
=&\begin{cases} \perp, \text{there is some } j \in \{1, \cdots, N\}, \text{such that } i_j \neq j \\ \mathsf{FE}^{(0)}.\mathsf{Enc}\left((m^{(0)}, \cdots, m^{(N)}), PRF(K, (\tau^{(1)}, \cdots, \tau^{(N)}))\right), \text{else} \end{cases}
\end{aligned}$$

(A reminder for readers: $(m^{(0)}, K)$ is a part of parameters of the function $f^{(0)}$, rather than a part of independent variables). Then randomly choose another function $f'^{(0)}$.

Step 6 For $i = 0, \cdots, N-1$, compute $c^{(i)} \leftarrow \mathsf{SKE}.\mathsf{Enc}(k^{(i)}, f^{(i)})$, $c'^{(i)} \leftarrow \mathsf{SKE}.\mathsf{Enc}(k'^{(i)}, f'^{(i)})$, where $k^{(i)}$ and $k'^{(i)}$ are taken as keys.

Step 7 For $i = 0, \cdots, N-1$, by $\{c^{(i)}, c'^{(i)}\}$ extend the function $f^{(i)}$ into $U^{(i)}_{c^{(i)}, c'^{(i)}}$, as the following detail. First, extend the independent variable of $f^{(i)}$, $((k^{(0)}, k'^{(0)}), \cdots, (k^{(i-1)}, k'^{(i-1)}), (m^{(i+1)}, m'^{(i+1)}, 1, \tau^{(i+1)}, i+1), \cdots, (m^{(N)}, m'^{(N)}, 1, \tau^{(N)}, N))$, into the independent variable of $U^{(i)}_{c^{(i)}, c'^{(i)}}$, $((k^{(0)}, k'^{(0)}), \cdots, (k^{(i)}, k'^{(i)}), (m^{(i+1)}, m'^{(i+1)}, 1, \tau^{(i+1)}, i+1), \cdots, (m^{(N)}, m'^{(N)}, 1, \tau^{(N)}, N))$, so that the independent variable of $U^{(i)}_{c^{(i)}, c'^{(i)}}$ is just the plaintext of $\mathsf{FE}^{(i)}.\mathsf{Enc}$ (See Table 2). Second, $U^{(i)}_{c^{(i)}, c'^{(i)}}$ is shown in Table 3.

Step 8 For $i = 0, \cdots, N-1$, construct the functional decryption key of the function $U^{(i)}_{c^{(i)}, c'^{(i)}}$ for $\mathrm{FE}^{(i)}$, and denote such decryption key by $M^{(i)}$.

Step 9 For $i = 0, \cdots, N$, take $M^{(i)}$ as corresponding ciphertext of $m^{(i)}$ for the frame (A reminder for readers: $M^{(N)}$ is in fact corresponding ciphertext of an extended plaintext for $\mathrm{FE}^{(N)}$, and the extended plaintext includes $m^{(N)}$. For $i = 0, \cdots, N-1$, $M^{(i)}$ is in fact a functional decryption key for $\mathrm{FE}^{(i)}$, and the shape of the function is related to $m^{(i)}$). Submit ($M^{(0)}$, $M^{(1)}$,$\cdots$, $M^{(N-1)}$, $M^{(N)}$) to the decryptor.

Operations of the system (Constructing functional decryption key)

Construct the functional decryption key of original function f for $\mathrm{FE}^{(0)}$, and take it as the functional decryption key of f for the frame, to be submitted to the decryptor.

Table 3. Function $U^{(i)}_{c^{(i)},c'^{(i)}}$

$U^{(i)}_{c^{(i)},c'^{(i)}}((k^{(0)},k'^{(0)}),\cdots,(k^{(i)},k'^{(i)}),(m^{(i+1)},m'^{(i+1)},1,\tau^{(i+1)},i+1),\cdots,(m^{(N)},m'^{(N)},1,\tau^{(N)},N))$
1.If $k^{(i)}\neq\perp$, compute $f^{(i)}\leftarrow\mathsf{SKE.Dec}(k^{(i)},c^{(i)})$ and output $f^{(i)}((k^{(0)},k'^{(0)}),\cdots,(k^{(i-1)},k'^{(i-1)}),(m^{(i+1)},m'^{(i+1)},1,\tau^{(i+1)},i+1),\cdots,(m^{(N)},m'^{(N)},1,\tau^{(N)},N))$.
2.Else, if $k'^{(i)}\neq\perp$, compute $f'^{(i)}\leftarrow\mathsf{SKE.Dec}(k'^{(i)},c'^{(i)})$ and output $f'^{(i)}((k^{(0)},k'^{(0)}),\cdots,(k^{(i-1)},k'^{(i-1)}),(m^{(i+1)},m'^{(i+1)},1,\tau^{(i+1)},i+1),\cdots,(m^{(N)},m'^{(N)},1,\tau^{(N)},N))$.
3.Else, output $\perp$.

Operations of the decryptor (Decryption)

First, it can be seen that $M^{(N)}$ is the first row of Table 2.

For FE$^{(N)}$, use $M^{(N-1)}$ acting on the first row of Table 2, to obtain the second row of Table 2 (as long as $k^{(N-1)}\neq\perp$).

For FE$^{(N-1)}$, use $M^{(N-2)}$ acting on the second row of Table 2, to obtain the third row of Table 2 (as long as $k^{(N-2)}\neq\perp$).

$\cdots\cdots$

For FE$^{(1)}$, use $M^{(0)}$ acting on the next last row of Table 2, to obtain the last row of Table 2 (as long as $k^{(0)}\neq\perp$).

Finally for FE$^{(0)}$, use the functional decryption key submitted by the system, acting on the last row of Table 2, to obtain $f(m^{(0)},\cdots,m^{(N)})$.

Because the shapes of $\{f^{(0)},\cdots,f^{(N-1)}\}$ include some information about $(m^{(0)},\cdots,m^{(N-1)})$, extended functions $\{U^{(0)}_{c^{(0)},c'^{(0)}},\cdots,U^{(N-1)}_{c^{(N-1)},c'^{(N-1)}}\}$ are needed to hide the shapes of $\{f^{(0)},\cdots,f^{(N-1)}\}$ from the decryptor. It can be seen that Table 3 and Table 1 are essentially same, which are from FPFE transformation [48].

3.4 AJ15 IO frame [10]

Suppose the function which needs to be obfuscated is $g(x_1,\cdots,x_N)$, where each entry x_i of the independent variable $(x_1,\cdots,x_N)$ is a bit variable. Now MIFE frame in subsection 3.3 is transformed into an IO frame. First, the encryptor and the system of MIFE frame are combined to become encoding-side of IO frame, and the decryptor of MIFE frame becomes decoding-side of IO frame. Second, all symbols of subsection 3.3 are still used in this subsection, but $(m^{(0)},m^{(1)},\cdots,m^{(N)})$ and f should be re-defined as such: $m^{(0)}=g$, $(m^{(1)},\cdots,m^{(N)})=(x_1,\cdots,x_N)$, and $f(m^{(0)},m^{(1)},\cdots,m^{(N)})=f(g,x_1,\cdots,x_N)=g(x_1,\cdots,x_N)$. In other words, f is taken as a public function group, while g is a secret function of such public group. The IO frame is described as follows.

Operations of encoding-side (Obfuscation)

Step 1 For $i=1,\cdots,N$, $m^{(i)}=0,1$, under MIFE frame compute corresponding ciphertext $M^{(i)}$ of $m^{(i)}$. More detailedly, when $m^{(i)}=0$ the corresponding ciphertext is denoted by $M^{(i)}_0$, and when $m^{(i)}=1$ the corresponding ciphertext is denoted by $M_1^{(i)}$ (A reminder for readers: these operations are under MIFE frame the operations of the encryptor).

Step 2 Under MIFE frame compute corresponding ciphertext $M^{(0)}$ of $m^{(0)}$ (A reminder for readers: such operation is under MIFE frame the operation of the encryptor).

Step 3 Under MIFE frame compute the functional decryption key of f, $key_{MIFE(f)}$ (A reminder for readers: such operation is under MIFE frame the operation of the system).

Step 4 Take $\{key_{MIFE(f)}$, $M^{(0)}$, $(M_0^{(1)}, M_1^{(1)})$, $\cdots$, $(M_0^{(N)}, M_1^{(N)})\}$ as the obfuscation of the function g, to be submitted to decoding-side (A reminder for readers: decoding-side should know that $M_0^{(i)}$ corresponds $m^{(i)} = 0$, and $M_1^{(i)}$ corresponds $m^{(i)} = 1$).

Operation of decoding-side (Computation)

Arbitrarily choose the value of the independent variable $(x_1, \cdots, x_N)$, let $(m^{(1)}, \cdots, m^{(N)}) = (x_1, \cdots, x_N)$, take corresponding $(M^{(1)}, \cdots, M^{(N)})(= (M^{(1)}_{m^{(1)}}, \cdots, M^{(N)}_{m^{(N)}}))$ of such $(m^{(1)}, \cdots, m^{(N)})$, and, by $\{key_{MIFE(f)}$, $M^{(0)}$, $M^{(1)}$, $\cdots$, $M^{(N)}\}$, under MIFE frame compute $f(m^{(0)}, m^{(1)}, \cdots, m^{(N)})(= g(x_1, \cdots, x_N))$ (A reminder for readers: decoding-side can repeat above operations, that is, choose different values of $(x_1, \cdots, x_N)$ and compute corresponding values of $g(x_1, \cdots, x_N)$).

3.5 Several explanations for AJ15 IO frame

AJ15 IO frame separates the independent variable into parts bitwise rather than a multi-bits part, for the purpose of decreasing the number of texts submitted to decoding-side. A part of the independent variable including h bits needs 2^h related texts to be submitted, while h parts with a single bit for each only need totally $2h$ related texts.

AJ15 IO frame takes g as $m^{(0)}$ rather than $m^{(i)}$ for $i = 1, \cdots, N$, for the purpose of computing the text including g as late as possible, in favour of protecting the shape of g.

To guarantee the effectiveness, AJ15 IO frame requires $\{\mathrm{FE}^{(0)}$, $\mathrm{FE}^{(1)}, \cdots$, $\mathrm{FE}^{(N)}\}$ to be compact FE schemes (CFE).

3.6 Cryptanalysis of AJ15 IO frame under RWB

Under RWB requirement, each of $\{M^{(0)}, (M_0^{(1)}, M_1^{(1)})$, $\cdots$, $(M_0^{(N)}, M_1^{(N)})\}$ is a fixed value rather than a black box, so the secret random numbers used for constructing each of them (including the encryption key of related FE scheme) should be changed into secret fixed numbers. The detailed observations are as follows. When $m^{(N)}$ is fixed, $\{(k^{(0)}, k'^{(0)})$, $\cdots$, $(k^{(N-1)}, k'^{(N-1)})$, $(m'^{(N)}, \tau^{(N)})$, the encryption key of $\mathrm{FE}^{(N)}$ $\}$ should be fixed (See the first row of Table 2). For $i = 1, \cdots, N-1$, when $m^{(i)}$ is fixed, $\{m'^{(i)}$, $\tau^{(i)}$, the encryption key of $\mathrm{FE}^{(i)}\}$ should be fixed (See the constructing procedure of $M^{(i)}$, operations of the encryptor step 4-8, subsection 3.3. In this case $M^{(i)}$ is a functional decryption key for $\mathrm{FE}^{(i+1)}$, and the parameter of the function includes $\{m'^{(i)}$, $\tau^{(i)}$, the encryption key of $\mathrm{FE}^{(i)}\})$. $\{K$, the encryption key of $\mathrm{FE}^{(0)}$ $\}$ should be constants (Because $m^{(0)} = g$ is fixed, $M^{(0)}$ is a functional decryption key for $\mathrm{FE}^{(1)}$, and the parameter of the function includes $\{K$, the encryption key of $\mathrm{FE}^{(0)}\})$. These observations are just Proposition 3.1.

Proposition 3.1 *In AJ15 IO frame satisfying RWB,*

(1) Secret random numbers $\{(k^{(0)}, k'^{(0)}), \cdots, (k^{(N-1)}, k'^{(N-1)}), (m'^{(N)}, \tau^{(N)})$, the encryption key of $FE^{(N)}$ $\}$ are only functions of $m^{(N)}$.

(2) For $i = 1, \cdots, N-1$, secret random numbers $\{m'^{(i)}, \tau^{(i)}$, the encryption key of $FE^{(i)}\}$ are only functions of $m^{(i)}$.

(3) $\{K$, the encryption key of $FE^{(0)}\}$ are only secret constants.

Besides secret random numbers of IO frame, encryption algorithms of FE schemes, $\{\mathsf{FE}^{(0)}.\mathsf{Enc}$, $\mathsf{FE}^{(1)}.\mathsf{Enc}$, $\cdots$, $\mathsf{FE}^{(N)}.\mathsf{Enc}\}$, may have other random numbers, which should be fixed as $(m^{(1)}, \cdots, m^{(N)})$ fixed. The detailed observations are as follows. When $m^{(N)}$ is fixed, random number of $\mathsf{FE}^{(N)}.\mathsf{Enc}$ (if exists) should be fixed (See the first row of Table 2). For $i = 1, \cdots, N-1$, when $m^{(i)}$ is fixed, random number of $\mathsf{FE}^{(i)}.\mathsf{Enc}$ (if exists) should be fixed (See the constructing procedure of $M^{(i)}$, operations of the encryptor step 4-8, subsection 3.3. In this case $M^{(i)}$ is a functional decryption key for $\mathrm{FE}^{(i+1)}$, and random number of $\mathsf{FE}^{(i)}.\mathsf{Enc}$ is an implicit parameter of such function). Random number of $\mathsf{FE}^{(0)}.\mathsf{Enc}$ (if exists) should be constant (Because $m^{(0)} = g$ is fixed, $M^{(0)}$ is a functional decryption key for $\mathrm{FE}^{(1)}$, and random number of $\mathsf{FE}^{(0)}.\mathsf{Enc}$ is an implicit parameter of such function). These observations are just Proposition 3.2.

Proposition 3.2 *In AJ15 IO frame satisfying RWB,*

(1) Random number of $\mathsf{FE}^{(N)}.\mathsf{Enc}$ *(if exists) is only a function of $m^{(N)}$.*

(2) For $i = 1, \cdots, N-1$, random number of $\mathsf{FE}^{(i)}.\mathsf{Enc}$ *(if exists) is only a function of $m^{(i)}$.*

(3) Random number of $\mathsf{FE}^{(0)}.\mathsf{Enc}$ *(if exists) is only a constant.*

Now observe the protection of IO frame for g. Decoding-side obtains $\{M^{(0)}$,$(M_0^{(1)}$, $M_1^{(1)})$, $\cdots$, $(M_0^{(N)}$, $M_1^{(N)})\}$, where $\{(M_0^{(1)}$, $M_1^{(1)})$, $\cdots$, $(M_0^{(N)}$, $M_1^{(N)})\}$ and corresponding computations have no relation with g. $M^{(0)}$ is related to g, but the shape of g is hidden by FPFE transformation ([48] and construction of extended function $U^{(0)}_{c^{(0)},c'^{(0)}}$ in this paper, see subsection 3.3). Then decoding-side uses $M^{(0)}$ for $\mathrm{FE}^{(1)}$ to obtain

$$\begin{aligned}&\mathsf{FE}^{(0)}.\mathsf{Enc}\left((m^{(0)}, m^{(1)}, \cdots, m^{(N)}), PRF(K, \tau^{(1)}, \cdots, \tau^{(N)})\right)\\ =&\mathsf{FE}^{(0)}.\mathsf{Enc}\left((g, m^{(1)}, \cdots, m^{(N)}), PRF(K, \tau^{(1)}(m^{(1)}), \cdots, \tau^{(N)}(m^{(N)}))\right)\end{aligned} \quad (1)$$

Then he uses $key_{MIFE(f)}$ for FE$^{(0)}$ to obtain

$$\begin{aligned}
&f\left((m^{(0)}, m^{(1)}, \cdots, m^{(N)}), PRF(K, \tau^{(1)}(m^{(1)}), \cdots, \tau^{(N)}(m^{(N)}))\right) \\
=&f(g, m^{(1)}, \cdots, m^{(N)}) \\
=&g(m^{(1)}, \cdots, m^{(N)})
\end{aligned} \tag{2}$$

Notice that $\mathsf{FE}^{(0)}.\mathsf{Enc}$ in formula (1) is not an ordinary encryption algorithm of FE$^{(0)}$, but rather a special encryption algorithm with random number fixed. In other words, in this case $\mathsf{FE}^{(0)}.\mathsf{Enc}$ is a fixed function of the plaintext, called encryption function. Recall AJ15 [10] and LV16/Lin17 [19, 21] which have such conclusion: "If secure CFE exists, IO can be constructed". Above observations show that such conclusion is at least not strict, and a stricter conclusion should be Proposition 3.3.

Proposition 3.3 *(Revised conclusion of Theorem 5.2 of AJ15 [10]) If secure CFE encryption function exists, IO can be constructed.*

Another special point of AJ15 IO frame is $PRF(K, \tau^{(1)}(m^{(1)}), \cdots, \tau^{(N)}(m^{(N)}))$ in formulae (1) and (2). First, its position in $\mathsf{FE}^{(0)}.\mathsf{Enc}$ is plaintext position (rather than random number position). Second, the computation result by using the functional decryption key $key_{MIFE(f)}$ to be $f(g, m^{(1)}, \cdots, m^{(N)})$ has no relation with it. Now we call $PRF(K, \tau^{(1)}(m^{(1)}), \cdots, \tau^{(N)}(m^{(N)}))$ the redundant plaintext of $\{\mathsf{FE}^{(0)}.\mathsf{Enc}, key_{MIFE(f)}\}$, and the following Proposition 3.4 holds.

Proposition 3.4 *Replace $PRF(K, \tau^{(1)}(m^{(1)}), \cdots, \tau^{(N)}(m^{(N)}))$ by any value in formula (1), then it can still be correctly decrypted to obtain $f(g, m^{(1)}, \cdots, m^{(N)}) = g(m^{(1)}, \cdots, m^{(N)})$.*

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 [35,38,39,44–47] 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^* [38–41] 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 $\{\{\overline{c_{ij}^{***}}\}, T\}$.

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 $\{\{\overline{c_{ij}^{****}}\}, T\}$.

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 LV16/Lin17 CFE Algorithms Inserted into AJ15 IO Frame under RWB

5.1 LV16/Lin17 CFE Algorithms Are Taken as FE$^{(0)}$ of AJ15 IO Frame

Under RWB requirement, this section presents cryptanalysis of LV16/Lin17 CFE algorithms which are inserted into AJ15 IO frame. More detailedly, we consider the case where LV16/Lin17 CFE algorithms are taken as FE$^{(0)}$ of AJ15 IO frame (see subsection 3.6). In such case the plaintext $x = (g, m^{(1)}, \cdots, m^{(N)}, PRF(K, \tau^{(1)}(m^{(1)}), \cdots, \tau^{(N)}(m^{(N)})))$ (see formula (1)), where $PRF(K, \tau^{(1)}(m^{(1)}), \cdots, \tau^{(N)}(m^{(N)}))$ is the redundant plaintext of $\{\mathsf{FE}^{(0)}.\mathsf{Enc}, key_{MIFE(f)}\}$ (see formula (1) and Proposition 3.4), K is a constant (see (3) of Proposition 3.1). According to RWB requirement, random numbers $\{k, s, b\}$ used in LV16/Lin17 CFE algorithms should be changed into constants (see (3) of Proposition 3.2), and $b = 0$.

LV/Lin17 schemes [19, 21] didn't describe the use of the redundant plaintext, so we respectively take two understandings, natural understanding and alternative understanding, described in subsections 5.2 and 5.3.

5.2 Cryptanalysis of FE$^{(0)}$ under Natural Understanding

Because the function $c\left(g, m^{(1)}, \cdots, m^{(N)}, PRF(K, \tau^{(1)}(m^{(1)}), \cdots, \tau^{(N)}(m^{(N)}))\right) = f(g, m^{(1)}, \cdots, m^{(N)}) = g(m^{(1)}, \cdots, m^{(N)})$ has no relation with the redundant plaintext $PRF(K, \tau^{(1)}(m^{(1)}), \cdots, \tau^{(N)}(m^{(N)}))$, natural understanding is that its garbling and randomized encoding $\{c^*_i\}$, $\{c^{***}_{ij}\}$ ($\{c^{****}_{ij}\}$) all have no relation with the redundant plaintext $PRF(K, \tau^{(1)}(m^{(1)}), \cdots, \tau^{(N)}(m^{(N)}))$.

Now $\{c^{***}_{ij}\}$ are low-degree low-locality functions of $(g, m^{(1)}, \cdots, m^{(N)}, k, s, b)$, which are learnable, and g cannot be hidden. Detailedly, decoding-side of AJ15 IO frame (the decryptor of FE$^{(0)}$) gradually probes the information of g, in the following three steps.

The first step, take $\{c^{***}_{ij}\}$ as low-degree low-locality functions of $(m^{(1)}, \cdots, m^{(N)})$, while (g, k, s) as only fixed parameters of these functions, to compute shapes of $\{c^{***}_{ij}\}$. Each c^{***}_{ij} has relation with $O(1)$ bits of $(m^{(1)}, \cdots, m^{(N)})$, so $2^{O(1)}$ values of such $O(1)$ bits determine the shape of c^{***}_{ij}. These shapes are possible to directly leak g, because randomized encoding does not promise to hide the function when random numbers are changed into constants. At least we know that the shape of each c^{***}_{ij} presents $2^{O(1)}$ equations of $O(1)$ bits of (g, k, s).

The second step, compute shapes of $\{c_i^*\}$ by those of $\{c_{ij}^{***}\}$. This step is simple because it is randomized decoding with symbols. Notice that parameter s has been eliminated, with parameters (g, k) left.

The third step, compute the shape of $c(g, \cdot)$ by those of $\{c_i^*\}$. This step is simple because it is de-garbling with symbols. Notice that parameter k has been eliminated, with parameter g left. In other words, the shape of $g(\cdot)\,(= c(g, \cdot))$ is obtained. g is a general function rather than a simple one, so the obtained shape is not the original shape of the function g, but it is never an "unintelligent shape (obfuscated shape)".

Each of the above three steps can obtain some information of the shape of the original function g. So that the IO scheme is not secure.

5.3 Cryptanalysis of FE$^{(0)}$ under Alternative Understanding

Alternative understanding is that $\{c_i^*\}$, $\{c_{ij}^{***}\}$ ($\{c_{ij}^{****}\}$) do have relation with $PRF(K, \tau^{(1)}(m^{(1)}), \cdots, \tau^{(N)}(m^{(N)}))$, but do not with (k, s), that is, (k, s) is replaced by $PRF(K, \tau^{(1)}(m^{(1)}), \cdots, \tau^{(N)}(m^{(N)}))$. In other words, $PRF(K, \tau^{(1)}(m^{(1)}), \cdots, \tau^{(N)}(m^{(N)}))$ is a part of the plaintext of FE$^{(0)}$ in terms of definition, while the random number in garbling-RE of c in terms of functionality. At first look, such alternative structure is perfect, which makes random number of garbling-RE to be kept random without increasing the degree of graded encoding.

However, we find a weakness of such alternative structure, called "difference perceptibility". Suppose we take a fixed bit p of $PRF(K, \tau^{(1)}(m^{(1)}), \cdots, \tau^{(N)}(m^{(N)}))$. According to (3) of Proposition 3.2, graded encoding procedure for p should not include random number, so graded encoding value $\overline{p}$ of p is a fixed function of p. That is, when decoding-side of AJ15 IO frame (the decryptor of FE$^{(0)}$) finds two graded encoding values $\overline{p_1}$ and $\overline{p_2}$ of p not equal, he knows $p_1 + 1 = p_2$ without knowing the value of $\{p_1, p_2\}$.

"Difference perceptibility" makes decoding-side of AJ15 IO frame (the decryptor of FE$^{(0)}$) to obtain equally many equations. Our analysis is like the first step in the last subsection as follows. Take a c_{ij}^{***}. Suppose c_{ij}^{***} is related with a_1 bits of $(m^{(1)}, \cdots, m^{(N)})$, denoted by m^*. Suppose c_{ij}^{***} is related with a_2 bits of $PRF(K, \tau^{(1)}(m^{(1)}), \cdots, \tau^{(N)}(m^{(N)}))$, denoted by p^*. Suppose c_{ij}^{***} is related with a_3 bits of g, denoted by g^*. Another part of $(m^{(1)}, \cdots, m^{(N)})$ without m^* is denoted by m^{**}. When $(m^{(1)}, \cdots, m^{(N)}) = (0, \cdots, 0)$, corresponding value of p^* is denoted by p_0^* (the decryptor of FE$^{(0)}$ does not know p_0^*, but can obtain $\overline{p_0^*}$). The value of $\overline{g^*}$ can be obtained by the decryptor. Now with known $\{\overline{g^*}, \overline{p_0^*}\}$, operations of the decryptor of FE$^{(0)}$ are as follows.

For each value of m^*, choose 2^{a_2} values of m^{**}, compute $\overline{m^*}$ and corresponding $\overline{p^*} = \overline{p_0^* + \delta}$, where δ is an a_2-dimensional Boolean vector, and the decryptor of FE$^{(0)}$ can obtain δ (because of the above described "difference perceptibility"). The chosen 2^{a_2} values of m^{**} should satisfy such condition: to guarantee δ takes 2^{a_2} different values (because the length of m^{**} is far larger than a_2, and PRF is a pseudo-random function, such condition is easy to be satisfied). By $(\overline{g^*}, \overline{m^*}, \overline{p_0^* + \delta})$ and zero-testing tool obtain $c_{ij}^{***}(g^*, m^*, p_0^* + \delta)$, and obtain the equation of (g^*, p_0^*):

$$c_{ij}^{***} = c_{ij}^{***}(g^*, m^*, p_0^*{+}\delta).$$

When m^* takes all values of $\{0,1\}^{a_1}$, δ takes all values of $\{0,1\}^{a_2}$, the decryptor of FE$^{(0)}$ obtains $2^{a_1+a_2}$ equations of (g^*, p_0^*). On the other hand, (g^*, p_0^*) is $a_2 + a_3$-dimensional Boolean vector, so $2^{a_2+a_3}$ equations are enough to solve it. If $a_1 \geq a_3$, (g^*, p_0^*) is solved. Notice that bits of (g^*, p_0^*) not only appear in c_{ij}^{***}, but also other functions of randomized encoding function family, to make related equations of other functions easier to be solved, finally g is leaked.

5.4 A Supplement

There may be such an idea that for $\mathsf{FE}^{(0)}.\mathsf{Enc}$, the graded encoding value of a plaintext bit is related not only with the value of this bit, but also with values of other plaintext bits (it means that a plaintext bit only has two values, while its graded encoding has exponentially many values). Such structure avoids the restriction of (3) of Proposition 3.2. Can it be such?

Notice that the zero-testing tool decoding-side of AJ15 IO frame (the decryptor of FE$^{(0)}$) owns is fixed. “Each bit has exponentially many graded encoding values” & “fixed zero-testing tool”, which is a higher requirement for graded encoding, and such a scheme has not appeared.

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 block-wise 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. 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).
24. Gentry, C., Jutla, C.S., Kane, D.: Obfuscation Using Tensor Products. In: Electronic Colloquium on Computational Complexity (ECCC), vol. 25 (2018).
25. Lin, H., Matt, C.: Pseudo Flawed-Smudging Generators and Their Application to Indistinguishability Obfuscation. Cryptology ePrint Archive, Report 2018/646 (2018).
26. 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
27. 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
28. 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
29. 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
30. 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
31. 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
32. 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).
33. 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
34. 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)
35. 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.

36. 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
37. 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`
38. 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).
39. 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
40. Applebaum, B., Ishai, Y., Kushilevitz, E.: Cryptography in nc0. In: Proceedings of the 45th FOCS, pp. 166-175 (2004).
41. 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
42. 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.
43. 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.
44. Yao, A.C.: Protocols for secure computations (extended abstract). In: Proceedings of the 23th FOCS, pp. 160-164 (1982).
45. Yao, A.C.: How to generate and exchange secrets (extended abstract). In: Proceedings of the 27th FOCS, pp. 162-167 (1986).
46. 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.
47. Gentry, C., Gorbunov, S., Halevi, S., Vaikuntanathan, V., Vinayagamurthy, D.: How to compress (reusable) garbled circuits. In: IACR eprint 2013/687.
48. Brakerski, Z., Segev, G.: Function-private functional encryption in the private-key setting. In: Journal of Cryptology, 31, 202-225.

Appendix A

Suppose that in randomized encoding schemes secret random numbers are changed into secret fixed numbers, and shapes of component functions are known. The leakage of the shape of original Boolean function is as the follow.

Randomized encoding schemes IK00/IK02/AIK04 [38–40] still have some abilities to hide original Boolean function, but far from protecting the shape of original Boolean function, and cannot promise that. More importantly, the premise of such hiding ability is that decoding-side knows nothing about the original Boolean function. For IO application, decoding-side knows that the original Boolean function is from a public function group, therefore the hiding is more fragile.

Randomized encoding scheme AIK06 [41] completely reveals the shape of original Boolean function.