Notes on Reusable Garbling[⋆]

Yupu Hu[1], Siyue Dong[1], Baocang Wang[1], and Jun Liu[1]

State Key Laboratory of Integrated Services Networks, Xidian University, Xi'an 710071, China
yphu@mail.xidian.edu.cn; 359442088@qq.com; bcwang@xidian.edu.cn; jliu6@stu.xidian.edu.cn

Abstract. Garbling is a cryptographic primitive which has many applications. It is mainly used for scenes of limited authority, such as multi-party computation (MPC), attribute-based encryption (ABE), functional encryption (FE), indistinguishability obfuscation (IO), etc. Garbling schemes before 2013 are of one-time garbling. Goldwasser et al and Agrawal presented a reusable garbling scheme, which made use of a symmetric encryption scheme and an FE scheme as the components.

In this paper we discuss the validity and the efficiency of reusable garbling scheme. We present the following three notes on the scheme.

(1) Reusable garbling scheme does not provide new applications, and it is still a one-time garbling scheme.

(2) Even reusable garbling scheme is taken as a one-time garbling scheme, sometimes it is not usable. More detailedly, it can only be used for Basic Scene 2, and cannot be used for Basic Scene 1. For example, it cannot be used for MPC.

(3) Even reusable garbling scheme is taken as a one-time garbling scheme used for Basic Scene 2, there is no evidence to show that its efficiency is better than a former one-time garbling scheme.

Keywords: garbling · functional encryption (FE) · fully homomorphic encryption (FHE) · attribute-based encryption (ABE)

1 Preliminaries: reusable garbling and our work

Garbling is a cryptographic primitive which has many applications, and is mainly used for scenes of limited authority. Garbling is firstly used for multi-party computation (MPC) [1, 2]. Then it is used for attribute-based encryption (ABE) and functional encryption (FE) [3, 4]. It is interesting that garbling and FE are usually

[⋆] Supported by National Natural Science Foundations of China (61972457, U19B2021); Key Research and Development Program of Shaanxi (2020ZDLGY08-04); Innovation Scientists and Technicians Troop Construction Projects of Henan Province.

the ground structure of each other. Garbling is also used for indistinguishability obfuscation (IO) [5–7]. Notice that garbling and obfuscation have similar functionality, but the latter is defined as more powerful.

1.1 Two basic scenes of garbling

All applications of garbling are based on the following two basic scenes, and are combinations or modifications of these two basic scenes.

Basic Scene 1 Alice knows Boolean circuit f, while Bob knows the value of Boolean variable x. Alice encodes f into $\overline{f}$, and encodes each value of X, the range of the variable (More detailedly, for each entry X_i of the range $X = (X_1, \cdots, X_n)$, respectively encodes 0 and 1 into $\overline{X_{i,0}}$ and $\overline{X_{i,1}}$. Thus the range X is encoded into $\overline{X} = \left(\overline{X_{1,0}}, \overline{X_{1,1}}, \cdots, \overline{X_{n,0}}, \overline{X_{n,1}}\right)$). Then Alice sends $\overline{f}$ to Bob, and sends $\overline{X}$ to Bob by "one-out-two oblivious transfer" (More detailedly, for two codes $\overline{X_{i,0}}$ and $\overline{X_{i,1}}$ of each entry X_i, Bob can choose to receive only one code). Bob chooses partial codes $\overline{x}$ from $\overline{X}$ according to the value of x. Then, "correctness" means that Bob can correctly compute $f(x)$ from $\left(\overline{x}, \overline{f}\right)$; "privacy"means that (1) Alice knows nothing about x, and (2) Bob knows nothing about f except the value of $f(x)$.

Basic Scene 2 Alice knows Boolean circuit f and the value of Boolean variable x. Alice encodes f into $\overline{f}$, and encodes x into $\overline{x}$. Alice sends $\left(\overline{x}, \overline{f}\right)$ to Bob. Then, "correctness" means that Bob can correctly compute $f(x)$ from $\left(\overline{x}, \overline{f}\right)$; "privacy" means that Bob knows nothing about (x, f) except the value of $f(x)$.

We have following three notes on above two basic scenes.

Note 1 "One-out-two oblivious transfer" in Basic Scene 1 is easy to be realized.

Note 2 Basic Scene 1 is just the ground structure of multi-party computation. Here we take the example of two users U_1 and U_2, suppose U_1 knows x_1, U_2 knows x_2, both U_1 and U_2 know the Boolean circuit $C(\cdot, \cdot)$. They want to compute $C(x_1, x_2)$, while U_1 hopes to hide x_1, U_2 hopes to hide x_2. Then, firstly take $f(\cdot) = C(x_1, \cdot)$, U_1 and U_2 take the roles of "Alice" and "Bob" in Basic Scene 1 respectively, so that U_2 computes $f(x_2) = C(x_1, x_2)$. Secondly take $f(\cdot) = C(\cdot, x_2)$, U_1 and U_2 exchange the roles, so that U_1 computes $f(x_1) = C(x_1, x_2)$. By taking Basic Scene 1 twice, both U_1 and U_2 obtain $C(x_1, x_2)$ (correctness), U_1 does not know x_2 while U_2 does not know x_1 (privacy).

Note 3 Basic Scene 2 cannot be the ground structure of multi-party computation. In Basic Scene 2, the knowledge of one side contains that of another side, which does not satisfy the scene assumption of multi-party computation.

1.2 Reusable garbling

Garbling schemes before 2013 are of one-time garbling [1, 2, 8, 9]. That is, encoding once (for Basic Scene 1 or Basic Scene 2) can only compute the combination $f(x)$ of one circuit f and one value of x, otherwise the privacy cannot be satisfied. Goldwasser et al [3] and Agrawal [4] presented a reusable garbling scheme, which made use of a symmetric encryption scheme and a functional encryption scheme as the components. From literal understanding, "reusable garbling" seems to be that encoding once can be repeatedly used to compute the combination $f(x)$ of one circuit f and any values of x.

1.3 Our work

In this paper we discuss the validity and the efficiency of reusable garbling scheme. We present the following three notes on the scheme.

(1) Reusable garbling scheme does not provide new applications, therefore it is still a one-time garbling scheme.

(2) Even reusable garbling scheme is taken as a one-time garbling scheme, sometimes it is not usable. More detailedly, it can only be used for Basic Scene 2, and cannot be used for Basic Scene 1. For example, it cannot be used for multi-party computation (MPC).

(3) Even reusable garbling scheme is taken as a one-time garbling scheme used for Basic Scene 2, there is no evidence to show that its efficiency is better than a former one-time garbling scheme [9].

In the following, we explain more details of our work.

To show the note (1), we point our that reusable garbling scheme only satisfies a weaker standard: encoding of f can be repeatedly used to compute $f(x)$ for different values of x, but encoding of variable (range or value) can only be used once, otherwise privacy cannot be satisfied. More than this, encoding of the variable is not independent of f, but rather needs information of f (For example, k_0, see subsection 3.1).

To show the note (2), we point out that in reusable garbling scheme, either the knowledge of Alice contains that of Bob, or just opposite, which does not satisfy the scene assumption of Basic Scene 1.

To show the note (3), we look into lower structure and ground structure of reusable garbling scheme (We pointed [11] that lower/ground structure of Agrawal [4] is invalid, so that in this paper we only look into lower/ground structure of Goldwasser et al [3]). Based on this, we compare reusable garbling scheme and an existing one-time garbling scheme [9]. We show that, if the former does not use "bootstrapping" and "modular switching" operations of fully homomorphic encryption technique, the computation complexity of the former is far larger than that of the latter; if the former uses "bootstrapping" or "modular switching" operations, the comparison is very complicated, but there is no evidence that the former has a smaller computation complexity than the latter.

We also consider a modified reusable garbling scheme, which replaces the FE scheme with "multi-input FE" [12]. After such modification, the scheme can achieve truly reusable garbling. However, multi-input FE takes IO as its ground structure.

2 BHR12: a major one-time garbling scheme [9]

2.1 General expression of Boolean circuit

We know a polynomial-time-computable Boolean circuit cannot be expressed by Algebraic Normal Form, because the number of terms may be super-polynomially large. The most reasonable expression may be describing {two input bits, Boolean operation, one output bit} of each step, according to the computing sequence. Suppose the set of Boolean operations is $\{+, \times\}$, where " + " is bit-exclusive-or, " × " is bit-and. Then n-dimensional Boolean circuit $f(x)$ is expressed as the follow.

- Input: $x = (x_1, x_2, \cdots, x_n), x_{n+1}$. Where $x_{n+1} = 1$ is constant.
- Computation: For $i = n+2, n+3, \cdots, N$, sequentially let $x_i \leftarrow x_{a(i)} O_i x_{b(i)}$, where $a(i) \in \{1, 2, \cdots, i-1\}$, $b(i) \in \{1, 2, \cdots, i-1\}$, $a(i) < b(i)$, $O_i \in \{+, \times\}$.
- Output: x_N ($x_N = f(x)$)

We call

(1) $\{(x_1, x_2, \cdots, x_n), x_{n+1}, (x_i, a(i), b(i), O_i), i = n+2, n+3, \cdots, N\}$ the general expression of Boolean circuit $f(x)$.

(2) N the circuit size of f.

(3) $\{(a(i), b(i)), i = n+2, n+3, \cdots, N\}$ the circuit topology of f.

(4) $\{O_i, i = n+2, n+3, \cdots, N\}$ the Boolean operation sequence of f.

2.2 BHR12 garbling scheme for Basic Scene 1

Take two public probability distributions X_0 and X_1, which are completely distinguishable. Take a public symmetric encryption algorithm $E(\cdot, \cdot)$, where the two positions are sequentially plaintext position and secret key position. The corresponding decryption algorithm is $D(\cdot, \cdot)$, where the two positions are sequentially ciphertext position and secret key position.

Encoding stage (Alice) Step 1 (Random definition): For each $i = 1, 2, \cdots, N-1$, randomly choose $j \in \{0, 1\}$, and define $X_{i,0} = X_j$, $X_{i,1} = X_{1-j (\bmod 2)}$. Define $X_{N,0} = X_0$, $X_{N,1} = X_1$.

Step 2 (Sampling): For each $i = 1, 2, \cdots, N, j \in \{0, 1\}$, sample $\overline{x_{i,j}} \leftarrow X_{i,j}$.

Step 3 (Encoding the variable range): Let $\overline{X} = (\overline{x_{1,0}}, \overline{x_{1,1}}, \cdots, \overline{x_{n,0}}, \overline{x_{n,1}}, \overline{x_{n+1,1}})$. Then $\overline{X}$ is encoding of the variable range.

Step 4 (Encoding the circuit f): For $i = n+2, \cdots, N$, $(x_{a(i)}, x_{b(i)}) \in \{(0, 0), (0, 1), (1, 0), (1, 1)\}$, if $x_{a(i)} O_i x_{b(i)} = u$, let

$$y_{i, x_{a(i)}, x_{b(i)}} = E\left(\overline{x_{i,u}}, \left(\overline{x_{a(i), x_{a(i)}}}, \overline{x_{b(i), x_{b(i)}}}\right)\right).$$

Then for each $i = n+2, \cdots, N$, there is a quartet of ciphertexts $\{y_{i,0,0}, y_{i,0,1}, y_{i,1,0}, y_{i,1,1}\}$. Take a random sort of the quartet, to obtain $\{\overline{y_{i,0,0}}, \overline{y_{i,0,1}}, \overline{y_{i,1,0}}, \overline{y_{i,1,1}}\}$ (The purpose of doing so is making Bob unable to obtain the information of any middle variable). Let $\overline{f} = \{(\overline{y_{i,0,0}}, \overline{y_{i,0,1}}, \overline{y_{i,1,0}}, \overline{y_{i,1,1}}), i = n+2, \cdots N\}$. Then $\overline{f}$ is the encoding of the circuit f.

Transmission stage (Alice $\rightarrow$ Bob) Alice sends $\overline{f}$ to Bob, and sends $\overline{X}$ to Bob by "one-out-two oblivious transfer" (That is, for each $i = 1, \cdots, n$, Bob can and only can choose one from $\{\overline{x_{i,0}}, \overline{x_{i,1}}\}$. Of course Bob can receive $\overline{x_{n+1,1}}$). Besides, Alice does not know Bob's choice.

Computation stage (Bob) First, notice that Bob knows the circuit topology of f. More detailedly, Bob knows a circuit class (universal circuit) which includes f, but does not know f. Bob obtains a part of codes of $\overline{X}$ according his x, and decrypts sequentially according to the topology of f. It is easy to see that, for each $i = n+2, \cdots, N$, Bob can only obtain one group from four groups $\{(\overline{x_{a(i),0}}, \overline{x_{b(i),0}}), (\overline{x_{a(i),0}}, \overline{x_{b(i),1}}), (\overline{x_{a(i),1}}, \overline{x_{b(i),0}}), (\overline{x_{a(i),1}}, \overline{x_{b(i),1}})\}$, so that he can only decrypt one value from two values $\{\overline{x_{i,0}}, \overline{x_{i,1}}\}$. Finally Bob decrypts one value from two values $\{\overline{x_{N,0}}, \overline{x_{N,1}}\}$. If the value is from X_0, $f(x) = 0$; if the value is from X_1, $f(x) = 1$.

2.3 BHR12 garbling scheme for Basic Scene 2 [9]

The unique difference of BHR12 scheme for Basic Scene 2 and for Basic Scene 1 is that Alice does not send $\overline{X}$ to Bob by "one-out-two oblivious transfer", but rather chooses directly $\overline{x}$ from $\overline{X}$ according to x, and sends $\{\overline{x}, \overline{f}\}$ to Bob. That means that the knowledge of Alice is $\{x, f\}$ and that of Bob is only $f(x)$.

3 Reusable garbling scheme [3, 4]

In this section we review reusable garbling scheme. We first state its structure, which means "reusable garbling" = "functional encryption" + "symmetric encryption". Then we state a lower structure, which means "functional encryption" =

"attribute-based encryption" + "fully homomorphic encryption" + "one-time garbling". Finally we state a ground structure, which means "one-time garbling" = "BHR12 scheme". By combining all statements, we know that "reusable garbing" = ABE + FHE + BHR12 + "symmetric encryption".

3.1 The structure

Take a public symmetric $E(\cdot,\cdot)$, where the two positions are sequentially plaintext position and secret key position, and corresponding decryption algorithm is $D(\cdot,\cdot)$. Take a functional encryption algorithm $E'(\cdot,\cdot)$, where the two positions are sequentially plaintext position and encryption key position. If encryption key is K, function is g, then corresponding decryption key is denoted by $S(K,g)$, and decryption will obtain the function value $g(m)$ of plaintext m, corresponding decryption algorithm is denoted by $D'(\cdot,\cdot)$. The structure has three stages: encoding, transmission, and computation.

Encoding stage (1) For Boolean function f, take a fixed symmetric key k_0, and compute $f^* = E(f,k_0)$. That is, take f as a bit string, and encrypt it by using symmetric key k_0.

(2) Take $f^{**}(x,k) = (D(f^*,k))\,(x)$. That is, $f^{**}(x,k)$ is a circuit as the follow: firstly compute $D(f^*,k)$ to obtain bit string F, then take F as a Boolean function $F(x)$ of the variable x. (It is easy to see that, if $k = k_0$, $f^{**}(x,k) = f^{**}(x,k_0) = f(x)$, while if $k \neq k_0$, $f^{**}(x,k)$ and $f(x)$ are independent of each other)

(3) Take a fixed encryption key K_0 of functional encryption algorithm, and compute corresponding decryption key $S(K_0,f^{**})$. Denote $S = S(K_0,f^{**})$.

(4) (Encoding the circuit f) Denote $\overline{f} = (f^*,S)$, $\overline{f}$ is encoding of original circuit f. Such $\overline{f}$ can be repeatedly used for computing $f(x)$ of different values of x, so that $\overline{f}$ is "reusable".

(5) (Encoding the variable range or value) For x, compute $\overline{x} = E'\left((x,k_0),K_0\right)$, $\overline{x}$ is the encoding of variable value x.

Transmission stage Send $(\overline{x}, \overline{f})$ to Bob. (The scene may be modified to such: Alice produces $\overline{f}$, Bob produces $\overline{x}$, Alice sends $\overline{f}$ to Bob)

Computation stage Bob obtains $(\bar{x}, \bar{f}) = (\bar{x}, f^*, S) = (\bar{x}, f^*, S(K_0, f^{**}))$. He computes

$$\begin{aligned} D'(\bar{x}, S) &= D'\left(\bar{x}, S\left(K_0, f^{**}\right)\right) \\ &= f^{**}\left(x, k_0\right) \\ &= f(x). \end{aligned}$$

It may be said that the computation of Bob does not use f^*, so that f^* needn't be sent to Bob. The fact is that functional decryption needs both the decryption key and the function. In other words, Bob's computation needs both S and f^{**}, and knowing f^* means knowing f^{**}.

3.2 A lower structure: about FE

Goldwasser et al [3] and Agrawal [4] respectively designed functional encryption schemes for the structure of reusable garbling. Because we point out [11] the invalidity of the functional encryption scheme of Agrawal [4], here we only review the scheme of Goldwasser et al [3]. This functional encryption scheme has such structure: "attribute-based encryption (ABE)" + "fully homomorphic encryption (FHE)" + "one-time garbing". Where the ABE is not original scheme, but the generalized version ABE$_2$. More detailedly, the ciphertext is not "decryptable if and only if the function value of the label equals to 1", but "a half is decryptable if the function value equals to 1, and another half is decryptable if the function value equals to 0". The task of generalizing ABE to ABE$_2$ is easy.

The key generation stage of FE has following six steps.

(1) Take an FHE scheme, where {FHE encryption key, corresponding decryption key} are taken as variable parameters.

(2) For f^{**} (see **Encoding stage** of subsection 3.1), construct f^{***}, the homomorphic function responding to f^{**} for FHE scheme. Now f^{***} is the function of FHE ciphertext.

(3) Express f^{***} in terms of entries, $f^{***}=\{f_1^{***},\cdots,f_\lambda^{***}\}$, where each f_i^{***} is a Boolean function of FHE ciphertext, λ is the number of entries (we know $\lambda=\lceil \log_2 q\rceil$, where q is the module of FHE scheme).

(4) Take λ ABE$_2$ schemes which are respectively denoted by ABE$_{2,1},\cdots,$ ABE$_{2,\lambda}$. All encryption keys of these ABE$_2$ schemes are taken as fixed parameters, and form K_0 (see **Encoding stage** of subsection 3.1).

(5) For each ABE$_{2,i}$, $i=1,\cdots,\lambda$, generate decryption key k_i according to Boolean function f_i^{***}. More detailedly, the functionality of k_i is as follows. Suppose the plaintext of ABE$_{2,i}$ is $\{p_0,p_1\}$. Decryption by k_i can only obtain p_1 if $f_i^{***}(label)=1$, and can only obtain p_0 if $f_i^{***}(label)=0$.

(6) $\{(f_i^{***},k_i),i=1,\cdots,\lambda\}$ are taken as fixed parameters, and form $S=S(K_0,f^{**})$ (see **Encoding stage** of subsection 3.1).

The encryption stage of FE has following six steps.

(1) Take (x,k_0) as FE plaintext (see **Encoding stage** of subsection 3.1).

(2) Generate {FHE encryption key, corresponding decryption key}, and encrypt (x,k_0) to a FHE ciphertext ψ. Then ψ is taken as the label (attribute) to be attached to each ABE$_2$ ciphertext.

(3) Compute $f^{***}(\psi)=(f_1^{***}(\psi),\cdots,f_\lambda^{***}(\psi))$. (We know that FHE decryption circuit containing decryption key can decrypt $f^{***}(\psi)$ to obtain $f^{**}(x,k_0)=f(x)$. However, FHE decryption circuit containing decryption key should be hidden, that is, FE decrypter shouldn't know FHE decryption key, otherwise the FE scheme is not secure). Denote $\psi^*=\{\psi_1^*,\cdots,\psi_\lambda^*\}=f^{***}(\psi)$.

(4) Suppose FHE decryption circuit containing decryption key is $fhe.dec(\cdot)$, we know it is only the function of $\psi^*=\{\psi_1^*,\cdots,\psi_\lambda^*\}$. Take a one-time garbling of the circuit $fhe.dec$, that is, take encoding $\overline{fhe.dec}$ of the circuit $fhe.dec$, and take encoding $\left\{\overline{\psi_{1,0}^*},\overline{\psi_{1,1}^*},\cdots,\overline{\psi_{\lambda,0}^*},\overline{\psi_{\lambda,1}^*}\right\}$ of the range of ψ^*.

(5) For $i=1,\cdots,\lambda$, encrypt $\left(\overline{\psi_{i,0}},\overline{\psi_{i,1}}\right)$ to obtain ABE$_{2,i}$ ciphertext c_i, with attached public label (ψ, FHE encryption key).

(6) Take $\left\{(c_1,\cdots,c_\lambda),\psi,\overline{fhe.dec}\right\}$ as the FE ciphertext.

The decryption stage of FE has following two steps.

(1) (Decryption of ABE_2) For $i = 1, \cdots, \lambda$, decrypt c_i to obtain $\mathrm{ABE}_{2,i}$ plaintext. More detailedly, obtain $\overline{\psi^*_{i,1}}$ if $\psi^*_i = 1$ and $\overline{\psi^*_{i,0}}$ if $\psi^*_i = 0$. Denote the decrypted value by $\overline{\psi^*_i}$.

(2) (Computation of one-time garbling) Compute

$$\begin{aligned}\overline{fhe.dec}\left(\overline{\psi^*_1}, \cdots, \overline{\psi^*_\lambda}\right) &= fhe.dec(\psi^*_1, \cdots, \psi^*_\lambda)\\ &= f^{**}(x, k_0)\\ &= f(x).\end{aligned}$$

3.3 A ground structure: about one-time garbling

BHR12 scheme [9] is a good candidate of one-time garbling for the FE scheme in subsection 3.2. BHR12 scheme does not restrict the function, as long as it is polynomial time computable, while other candidates restrict the depth of the function. Another benefit of BHR12 scheme is that it only uses symmetric encryption without special cryptographic tools, while other candidates need branching program or randomized matrix or something else.

4 Analysis of the validity of reusable garbling scheme

4.1 Analysis of reusability

Encoding $\overline{f}$ of circuit f can be repeatedly used for computing $f(x)$ of different values of x. But there is only encoding $\overline{x}$ of single value x, rather than encoding of variable range. That is, before each computation, encoding of a new value is needed. More than this, encoding of the variable value is not independent of f, but rather needs some information of f (For example, k_0, see subsection 3.1). The conclusion is that "encoding" as whole entity is not reusable, so that reusable garbling scheme does not provide new applications.

4.2 Analysis of scenes

We look into the structure (see subsection 3.1). Suppose Alice encodes the circuit f, so that Alice knows f.

If Alice encodes a variable value x, Alice knows x. In this case the knowledge of Alice is $\{x, f\}$, while that of Bob is only $f(x)$.

If Bob encodes a variable value x, he needs to know not only x but also k_0. On the other hand, Bob certainly knows f^* (because f^* is a part of $\overline{f}$), so that Bob immediately obtain f: $f = D(f^*, k_0)$. In this case the knowledge of Alice is only f, while that of Bob is $\{x, f\}$.

From all of the above, the knowledge of one side always contains that of another side, which does not satisfy the knowledge distribution of Basic Scene 1. Therefore, even reusable garbling scheme is taken as a one-time garbling scheme, at least it cannot be used for MPC.

5 Analysis of the efficiency of reusable garbling scheme

In this section reusable garbling scheme is taken as a one-time garbling scheme used for Basic Scene 2 (see section 1), and we compare it with BHR12 scheme [9].

Suppose the length of x is n, the length of f (as a bit-string) is n', the circuit size of f (as a Boolean circuit) is N. Suppose reusable garbling scheme and BHR12 scheme use same symmetric encryption scheme $\{E(\cdot,\cdot), D(\cdot,\cdot)\}$. Suppose the circuit size of $E(\cdot,\cdot)$ is N' (Notice that $E(\cdot,\cdot)$ is a multi-output Boolean circuit, and it is easy to generalize the concept of circuit size from single-output Boolean circuit to multi-output Boolean circuit). According to present situation of symmetric encryption technique, $D(\cdot,\cdot)$ has a circuit size a little larger than N'. Suppose the number of ABE$_2$ schemes for reusable garbling scheme is λ, and we know $\lambda = \lceil \log_2 q \rceil$, q is the module of FHE scheme.

5.1 Computation complexity of BHR12 garbling scheme

In the encoding stage of BHR12 scheme, major computations are $4(N-n-1)$ times using symmetric encryption algorithm $E(\cdot,\cdot)$. Other computations are $2N-1$ times of sampling and $N-n-1$ times of random sort of quartet.

In the computation stage of BHR12 scheme, the computations are $4(N-n-1)$ times using symmetric decryption algorithm $D(\cdot,\cdot)$, where $N-n-1$ times are successful decryption, while other $3(N-n-1)$ times are decryption failure.

5.2 Our two observations

The first observation is that, if reusable garbling scheme does not use "bootstrapping" and "modular switching" operations of FHE technique, generally we have $\lambda > 4N$. The reason is the follow.

Because the circuit size of f is N and that of $D(\cdot,\cdot)$ a little larger than N', the circuit size of f^{**} is a little larger than $N+N'$. In other words, f^{**} is sequentially about $N+N'$ Boolean operations. As a homomorphic function of f^{**}, f^{***} should be sequentially about $N+N'$ modular q operations, and $\{f^{**}, f^{***}\}$ should have corresponding operation types: a bit-exclusive-or corresponds to a modular-q-addition, while a bit-and corresponds to a modular-q-multiplication. Now we consider noise size accumulation of such $N+N'$ modular q operations. Suppose initial noise size is averagely e. According to the security of FHE scheme, generally e should be polynomially large, so that $\log_2 e \geq 8$ is reasonable. Another reasonable assumption is that, from such $N+N'$ modular q operations, about half are multiplications, another half are additions. Modular-q-addition operation will make the noise size larger, but the effect is limited, so we ignore it. Modular-q-multiplication makes the noise size which is about the product of original two noise sizes. $\frac{N+N'}{2}$ modular-q-multiplication operations will make the noise size which is about $e^{\frac{N+N'}{2}}$. On the other hand, a condition for correct decryption is that the module is larger than the double of the noise size, therefore $q > 2\cdot e^{\frac{N+N'}{2}}$. Finally we have

$$\lambda = \lceil \log_2 q \rceil \geq \frac{N+N'}{2}\cdot \log_2 e \geq 4(N+N') > 4N.$$

The second observation is that, if the reusable garbling scheme does not use "bootstrapping" and "modular switching" operations, the computation complexity of $\{\text{ABE}_2 \text{ encryption}, \text{ABE}_2 \text{ decryption}\}$ is larger than that of $(E(\cdot,\cdot), D(\cdot,\cdot))$. The reason is the follow.

Notice

(1) $f^{***} = \{f_1^{***}, \cdots, f_\lambda^{***}\}$.

(2) A Boolean operation of f^{**} corresponds to a modular q operation of f^{***}.

(3) For each $i = 1, \cdots, \lambda$, a modular q operation of f^{***} corresponds to far more than one Boolean operation of f_i^{***}. In other words, the circuit size of each f_i^{***} is far larger than that of f^{**}, which is a little large than $N + N'$.

(4) On the other hand, according to present situation of KP-ABE [13, 14], for each Boolean operation O of f_i^{***}, $\mathrm{ABE}_{2,i}$ decryption algorithm has a corresponding group of operations O', where the computation complexity of O' is far larger than that of O. We call O' the quasi-homomorphic operation of O.

From all of the above, $\{\mathrm{ABE}_2$ encryption, ABE_2 decryption$\}$ has the computation complexity far larger than $2(N + N')$, while $\{E(\cdot,\cdot), D(\cdot,\cdot)\}$ has the computation complexity a little larger than $2N'$.

5.3 Comparison: neither "bootstrapping" nor "modular switching"

Major computation of BHR12 scheme are $\{E(\cdot,\cdot), D(\cdot,\cdot)\}$ about 4N times. Major computation of reusable garbling scheme are $\{\mathrm{ABE}_2$ encryption, ABE_2 decryption$\}$ λ times, an FHE, and a one-time garbling. According to our two observations (see subsection 5.2), computation complexity of reusable garbling scheme is far larger than that of BHR12 scheme.

5.4 Comparison: with "bootstrapping" or "modular switching"

If reusable garbling scheme uses "bootstrapping" or "modular switching" operations of FHE technique, homomorphic function is $f^{****} = \{f_1^{****}, \cdots, f_{\lambda'}^{****}\}$ rather than $f^{***} = \{f_1^{***}, \cdots, f_\lambda^{***}\}$, where $\lambda' = \lceil \log_2 q' \rceil$, q' is a smaller module (That is, $q' < q$). In this case, $\{\mathrm{ABE}_2$ encryption, ABE_2 decryption$\}$ is for $\{f_{i'}^{****}, i' = 1, \cdots, \lambda'\}$ rather than for $\{f_i^{***}, i = 1, \cdots, \lambda\}$. f^{****} has less entries than f^{***}, therefore uses ABE_2 less times. But f^{****} has entries of larger circuit sizes, therefore each time of using ABE_2 has larger computation complexity.

From all of the above, the comparison is very complicated, but there is no evidence that BHR12 scheme has a larger computation complexity than reusable garbling scheme with "bootstrapping" or "Modular switching" operations.

6 About a modified scheme

Goldwasser et al [12] presented multi-input FE scheme. If such scheme replaces the FE scheme in the structure of reusable garbling (see subsection 3.1), the modified reusable garbling scheme can not only be used for Basic Scene 1, but also achieve stronger reusability of IO. More detailedly, by the feature of "dispersed encryption and centralized decryption", in the encryption stage of FE (subsection 3.1, step(5) of **Encoding stage**), $\overline{x} = \{E'(x, K_0), E'(k_0, K_0)\}$ rather than $\overline{x} = E'\left((x, k_0), K_0\right)$; in the decryption stage of FE (subsection 3.1, **Computation stage**), $D'(\overline{x}, S) = D'\left(\left(E'(x, K_0), E'(k_0, K_0)\right), S\right) = f(x)$.

After such modification, the task of computing $E'(x, K_0)$ is assigned to Bob, and computing $E'(k_0, K_0)$ to Alice. Then all contents Alice submits are reusable, and Bob can repeatedly compute for any times without contacting Alice each time.

However, such modified FE takes IO as its ground structure. If IO exists, itself is a powerful reusable garbling, and there is no need for using it to construct another reusable garbling system. Construction of reusable garbling [3, 4] avoids IO for the purpose of avoiding huge size and obscure security.

References

1. Yao, A, C.: Protocols for secure computations (extended abstract)[C]. In: Proceedings of the 23rd Annual Symposium on Foundations of Computer Science (SFCS 1982). IEEE, 1982: 160-164. [DOI: 10.1109/SFCS.1982.38]
2. Yao, A, C.: How to generate and exchange secrets (extended abstract)[C]. In: Proceedings of the 27th Annual Symposium on Foundations of Computer Science (SFCS 1986). IEEE, 1986: 162-167. [DOI: 10.1109/SFCS.1986.25]
3. Goldwasser, S., Kalai, Y., Popa, R, A., et al.: Reusable garbled circuits and succinct functional encryption[C]. In: Proceedings of the forty-fifth annual ACM symposium on Theory of computing. ACM, 2013: 555-564. [DOI: 10.1145/2488608.2488678]

4. 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
5. Gentry, C., Gorbunov, S., Halevi, S., et al.: How to compress (reusable) garbled circuits[J]. IACR Cryptology ePrint Archive, 2013: 2013/687.
6. Lin, H, J.: Indistinguishability obfuscation from constant-degree graded encoding schemes[C]. In: Advances in Cryptology-EUROCRYPT 2016. Springer Berlin Heidelberg, 2016: 28-57. [DOI: 10.1007/978-3-662-49890-3_2]
7. Lin, H, J.: Indistinguishability obfuscation from SXDH on 5-linear maps and locality-5 PRGs[C]. In: Advances in Cryptology-CRYPTO 2017. Springer Berlin Heidelberg, 2017: 599-629. [DOI: 10.1007/978-3-319-63688-7_20]
8. Ishai, Y., Kushilevitz, E.: Randomizing polynomials: A new representation with applications to round-efficient secure computation[C]. In: Proceedings of the 41st Annual Symposium on Foundations of Computer Science (SFCS 2000). IEEE, 2000: 294-304. [DOI: 10.1109/SFCS.2000.892118]
9. Bellare, M., Hoang, V.T., Rogaway, P.: Foundations of garbled circuits. In: Yu, T., Danezis, G., Gligor, V.D. (eds.) ACM CCS 2012, pp. 784–796. ACM Press, October 2012 [DOI: 10.1145/2382196.2382279]
10. Hu, Y, P., Liu, J., WANG, B, C.: Efficiency Analysis and Simplified Scheme of Reusable Garbling. Journal of Cryptologic Research, 2022, 9(1): 106-112.
11. Hu, Y, P., Liu, J., WANG, B, C., et al.: *P/poly* Invalidity of the Agr17 Functional Encryption Scheme. Cryptology ePrint Archive, Paper 2021/1442.`https://eprint.iacr.org/2021/1442`
12. Goldwasser, S., Gordon, S, D., Goyal, V., et al.: Multi-input Functional Encryption. In: Advances in Cryptology-EUROCRYPT 2014. Springer Berlin Heidelberg, 2014: 578-602. [DOI: 10.1007/978-3-642-55220-5_32]
13. Gorbunov, S., Vaikuntanathan, V., Wee, H.: Attribute based encryption for circuits. In: STOC 2013. pp. 545–554. ACM (2013). https://doi.org/ [DOI: 10.1145/2488608.2488677]
14. Boneh, D., Gentry, C., Gorbunov, S., Halevi, S., Nikolaenko, V., Segev, G., Vaikuntanathan, V., Vinayagamurthy, D.: Fully Key-Homomorphic Encryption, Arithmetic Circuit ABE and Compact Garbled Circuits. In: Nguyen, P.Q., Oswald, E. (eds) Advances in Cryptology – EUROCRYPT 2014. EUROCRYPT 2014. Lecture Notes in Computer Science, vol 8441. Springer, Berlin, Heidelberg. [DOI: 10.1007/978-3-642-55220-5_30]