P/poly Invalidity of the Agr17 Functional Encryption Scheme*

Yupu Hu[1], Jun Liu[1], Baocang Wang[1], Xingting Dong[1], and Yanbin Pan[2]

[1] State Key Laboratory of Integrated Services Networks, Xidian University, Xi'an 710071, China
yphu@mail.xidian.edu.cn; jliu6@stu.xidian.edu.cn; bcwang@xidian.edu.cn; xtdong67@163.com

[2] Key Laboratory of Mathematics Mechanization, NCMIS, Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing 100190, China
panyanbin@amss.ac.cn

Abstract. Functional encryption (FE) is an advanced topic in the research of cryptography, and the Agr17 FE scheme is one of the major FE schemes. It took the BGG+14 attribute-based encryption (ABE) scheme as a bottom structure, which was upgraded into a 'partially hiding predicate encryption' (PHPE) scheme and combined with a fully homomorphic encryption (FHE) scheme. However, there is a remaining problem, the implementation of the modulus reduction, in the Agr17 FE scheme. First, a modulus reduction is necessary for the polynomial-time computability of the scheme. Second, the detailed steps of the modulus reduction were absent in the scheme (including its conference version and full version). Instead, the authors only pointed out several reference works. The author's meaning seemed to be that the modulus reduction of the Agr17 FE scheme can be obtained by directly using or simply generalizing these reference works. Third, these reference works only described various modulus reductions of FHE schemes, without the hint of how to generalize them into the modulus reduction of FE schemes. Finally, any modulus reduction of FHE can not be simply generalized into the modulus reduction of the Agr17 FE scheme due to the following two facts: (1) The Agr17 FE scheme has two moduli, which are the modulus of the FHE ciphertext and of the ABE ciphertext, both are originally superpolynomial in size for processing $P/poly$ functions. (2) Both moduli need to be scaled down to polynomial size, and both of them need to be reduced to the same new modulus, otherwise, the correctness of the scheme will fail.

In this paper, we demonstrate that the Agr17 FE scheme is $P/poly$ invalid. More specifically, we show that, when processing $P/poly$ functions, the Agr17 FE scheme cannot be implemented again after its modulus reduction. To show the soundness of our demonstration, we present the statements in two stages. At the first stage, we show that the modulus

* 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.

reduction of the Agr17 FE scheme should be a double modulus reduction, which includes two modulus reductions for the FHE ciphertext and ABE ciphertext, respectively. This double modulus reduction has the following three key points: (1) The modulus reduction for the FHE ciphertext should be seen as a series of Boolean operations, and converted into 'attribute quasi-homomorphic operations'. (2) The modulus reduction for the ABE ciphertext is a learning-with-errors (LWE) -based modulus reduction, which is an ordinary modulus reduction. (3) The two modulus reductions should obtain the same new modulus, otherwise, the scheme would not be implemented again. At the second stage, we show that the modulus reduction for the ABE ciphertext will destroy the structure of ABE so that the subsequent decryption would not be executed. The reason lies in that the decryption of ABE is an LWE decryption with conditions rather than an ordinary LWE decryption, and the modulus reduction will destroy the conditions of decryption. Besides, to show such invalidity cannot be easily crossed by revising the scheme, we design two revised versions of the Agr17 scheme. The first revised version is a 'natural' revised version of the Agr17 scheme. The key point is to change the small modulus inner product into an arithmetic inner product, which can be obtained by the modulus inner product of the ABE ciphertext. The first revised scheme is valid, i.e., the decryption can be implemented correctly. However, the revised scheme is insecure because the decryptor knows much more secret information, and hence the scheme can be broken by collusion attacks with much less cost. The second revised version is an application of the GGH+13b verification circuit technology which transforms a $P/poly$ function into an NC^1 circuit. The second revised scheme is valid, but it is far from the design idea of the Agr17 scheme, and its function class is quite limited, that is, those functions which can be equally transformed from $P/poly$ into NC^1 by equal verification transformation, rather than any $P/poly$ functions.

Keywords: learning with errors · attribute-based encryption · functional encryption.

1 Introduction

1.1 Problems and Contributions

The scenario of functional encryption (FE) is that an encryptor transforms a plaintext into a ciphertext, and a decryptor can only transform the ciphertext into a function value of the plaintext (rather than the plaintext). FE is an advanced topic in the research of cryptography. Since Boneh et al. proposed the formal definition [1], FE has made great progress [2–20]. The Agr17 FE scheme is one of the major FE schemes. It took the BGG+14 attribute-based encryption (ABE) scheme as a bottom structure [21], which was upgraded into a 'partially hiding predicate encryption'

(PHPE) scheme and combined with a fully homomorphic encryption (FHE) scheme. Subsequent LLW21 FE scheme [20], which is of the more optimized sampling characteristics and the similar structure, is an improved work of the Agr17 FE scheme. In addition, the Agr17 FE scheme was cited by a dozen of works [22–32].

However, the implementation of the modulus reduction is a remaining problem in the Agr17 FE scheme.

First, a modulus reduction is necessary. The Agr17 FE scheme needs to traverse all possible errors, while the size of the errors has the same order of magnitude as the size of the modulus. Therefore, only by reducing the modulus to polynomial size, can the exhaustion of errors be completed in polynomial time, thus leading to a polynomial-time computable scheme.

Second, the detailed steps of the modulus reduction were absent in the scheme (including its conference version and full version). Instead, the authors only pointed out several reference works [33–35]. The author's meaning seemed to be that the modulus reduction of the Agr17 FE scheme can be obtained by directly using or simply generalizing these reference works.

Third, these reference works [33–35] only described various modulus reductions of FHE schemes, without the hint of how to generalize them into the modulus reduction of FE schemes. We have tried many methods to 'naturally' extend various modulus reductions of FHE to FE but all failed.

Finally, any modulus reduction of FHE can not be simply generalized into the modulus reduction of the Agr17 FE scheme due to the following two facts: (1) The Agr17 FE scheme has two moduli, which are the modulus of the FHE ciphertext (we call it an inner modulus or implicit modulus) and of the ABE ciphertext (we call it an outer modulus or explicit modulus), both are originally superpolynomially large for processing $P/poly$ function. Concretely, because both of the two moduli are employed to process $P/poly$ function, the sizes of the accumulated errors should be superpolynomially large. (2) Both moduli need to be reduced to polynomially large, and both of them need to be reduced to the same new modulus, otherwise, the correctness of the scheme will fail. The reason is that the restrictions on operations

when hiding attributes are extremely strict in the BGG+14 ABE scheme. Next, we will recount these two facts.

In this paper, we demonstrate that the Agr17 FE scheme is $P/poly$ invalid, i.e., this scheme cannot be executed for $P/poly$ functions. More specifically, this scheme cannot be implemented after its modulus reduction in the decryption phase. To show the soundness of our demonstration, we present the statements in two stages.

At the first stage, we show that a single modulus reduction is not feasible since the subsequent inner products will be impossible to achieve. Therefore, the modulus reduction of the Agr17 FE scheme should be a double modulus reduction, which includes two modulus reductions for the FHE ciphertext and ABE ciphertext, respectively. This double modulus reduction has the following three key points: (1) The modulus reduction for the FHE ciphertext should be seen as a series of Boolean operations, and converted into 'attribute quasi-homomorphic operations'. (2) The modulus reduction for the ABE ciphertext is a learning-with-errors (LWE)-based modulus reduction, which is an ordinary modulus reduction. (3) The two modulus reductions should obtain the same new modulus, otherwise, the scheme would not be performed further.

At the second stage, we show that the modulus reduction for the ABE ciphertext will destroy the structure of ABE so that the subsequent decryption would not be executed. Note that for a general LWE decryption, the modulus reduction for the ciphertext is feasible, although with some limitations, e.g., some values should be small. However, the decryption of the BGG+14 ABE scheme is an LWE decryption with conditions rather than an ordinary LWE decryption. The conditions are embedded into the structure of the ABE ciphertext. Consequently, the modulus reduction will destroy the structure of the ABE ciphertext, and hence the conditions of decryption.

Besides, to show such invalidity cannot be easily crossed by revising the scheme, we design two revised versions of the Agr17 scheme. The first revised version is a 'natural' revised version of the Agr17 scheme. The key point is to change the small modulus inner product into an arithmetic inner product, which can be obtained

by the modulus inner product of the ABE ciphertext. The first revised scheme is valid, i.e., the decryption can be implemented correctly. However, the revised scheme is insecure because the decryptor knows much more secret information, and hence the scheme can be broken by collusion attacks with much less cost. The second revised version is an application of the GGH+13b verification circuit technology which transforms a $P/poly$ function into an NC^1 circuit. The second revised scheme is valid, but it is far from the design idea of the Agr17 scheme, and its function class is quite limited, that is, those functions which can be equally transformed from $P/poly$ into NC^1 by equal verification transformation, rather than any $P/poly$ functions.

1.2 Organization

The remainder of the paper is organized as below. The BGG+14 ABE scheme, which is the bottom structure of the Agr17 FE scheme, is presented in Sect. 2. There are several annotations regarding the details of the BGG+14 scheme. Particularly, we explain why the modulus of this scheme should be superpolynomially large when processing $P/poly$ functions and the limitations on the operations when partially hiding attributes. In Sect. 3, the Agr17 FE scheme is described, including our annotations on the special processing idea about the modulus reduction. Sect. 4 introduces the core contribution of this paper, which is to demonstrate the $P/poly$ invalidity of the Agr17 FE scheme. First, we state that the modulus reduction should be a double modulus reduction when processing $P/poly$ functions, and we propose the technical details of the double modulus reduction. Then, we state that the modulus reduction for the ABE ciphertext will destroy the structure of ABE so that the subsequent operations would not be executed further. In Sect. 5, we present two revised versions of the Agr17 scheme, and state their weakness: The first revised version is insecure although the decryption can be executed correctly. The second revised version is far from the design idea of the Agr17 scheme and the function class is quite limited.

2 Bottom Structure of the Agr17 FE Scheme: BGG+14 ABE Scheme

2.1 Notations and Operations

Let (m, n, q) denote three positive integers such that $q = n^{\Theta(d_{max})}$ where q is prime, $m = n\lceil \log_2 q \rceil$, and d_{max} has been well explained. Let $\mathbb{Z}$ denote the set of integers. For two positive integers (m', m''), $(\mathbb{Z}^{m'}, \mathbb{Z}^{m' \times m''}, \mathbb{Z}_q^{m'}, \mathbb{Z}_q^{m' \times m''})$ have been well defined. Note that the output of the operation 'mod$q$' is within $\{\lceil -\frac{q}{2} \rceil, \lceil -\frac{q}{2} \rceil + 1, \ldots, \lceil \frac{q}{2} \rceil\}$, rather than $\{0, 1, \ldots, q-1\}$. For $a \in \mathbb{Z}_q$, the meanings of $\mathbf{A} \in \mathbb{Z}_q^{m' \times m''}$ and $a\mathbf{A} \in \mathbb{Z}_q^{m' \times m''}$ are clear. Let $\mathbf{G}$ denote the following special matrix

$$\mathbf{G} = \begin{pmatrix} 1\, 2 \cdots 2^{\lceil \log_2 q \rceil - 1} & & & \\ & 1\, 2 \cdots 2^{\lceil \log_2 q \rceil - 1} & & \\ & & \ddots & \\ & & & 1\, 2 \cdots 2^{\lceil \log_2 q \rceil - 1} \end{pmatrix} \in \mathbb{Z}_q^{n \times m}.$$

For any $\alpha \in \mathbb{Z}_q$, there is a unique Boolean matrix $\mathbf{G}^{(\alpha)} \in \mathbb{Z}^{m \times m}$ such that

$$\alpha \mathbf{G} = \mathbf{G}\mathbf{G}^{(\alpha)} (\mathrm{mod} q).$$

For any $\mathbf{B} \in \mathbb{Z}_q^{n \times m}$, there is a unique Boolean matrix $\mathbf{G}^{(\mathbf{B})} \in \mathbb{Z}^{m \times m}$ such that

$$\mathbf{B} = \mathbf{G}\mathbf{G}^{(\mathbf{B})} (\mathrm{mod} q).$$

Then, the following three algorithms are well known.

- $\mathsf{TrapGen}(n, m, q)$: Input (n, m, q) and output $(\mathbf{A}, \mathbf{T})$, where $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ is a uniform matrix, $\mathbf{T} \in \mathbb{Z}^{m \times m}$ is a small Gaussian matrix, $\mathbf{A}\mathbf{T} = \mathbf{0} \in \mathbb{Z}_q^{n \times m}$, and $\mathbf{T}$ is of full rank. $\mathbf{T}$ is not of full rank regarding the modulus q. $\mathbf{T}$ is called a trapdoor of $\mathbf{A}$.
- $\mathsf{Encode}(\mathbf{A}, \mathbf{s})$: Input $(\mathbf{A}, \mathbf{s}) \in \mathbb{Z}_q^{n \times m} \times \mathbb{Z}_q^n$ and output $\boldsymbol{\psi} = \mathbf{A}^T\mathbf{s} + \mathbf{e} \in \mathbb{Z}_q^m$, where $\mathbf{e} \in \mathbb{Z}^m$ is a small Gaussian vector. $\mathbf{s}$ is the encoded vector, $\boldsymbol{\psi}$ is the encoding of $\mathbf{s}$, and $\mathbf{e}$ is the error vector. We say $\boldsymbol{\psi} = \mathsf{Encode}(\mathbf{A}, \mathbf{s})$.

– $\mathsf{ReKeyGen}(\mathbf{A}, \mathbf{B}, \mathbf{T}, \mathbf{D})$: Input $(\mathbf{A}, \mathbf{B}, \mathbf{T}, \mathbf{D})$ and output $\mathbf{R}$, where $\mathbf{A}, \mathbf{B} \in \mathbb{Z}_q^{n\times m}$ are uniform matrices, $\mathbf{T} \in \mathbb{Z}^{m\times m}$ is a trapdoor of $\mathbf{A}$, $\mathbf{R} \in \mathbb{Z}^{2m\times m}$ is a small Gaussian matrix, and $\mathbf{D} = [\mathbf{A}, \mathbf{B}]\mathbf{R} \in \mathbb{Z}_q^{n\times m}$. In fact,

$$\mathbf{R} = \begin{bmatrix} \mathbf{R}_0 \\ \mathbf{R}_1 \end{bmatrix}, \mathbf{R}_i \in \mathbb{Z}^{m\times m}, i = 0, 1,$$

then $\mathbf{R}_1$ is the pre-sampled matrix, and $\mathbf{R}_0$ is the co-sampled matrix. The trapdoor matrix $\mathbf{T}$ satisfies that $\mathbf{A}\mathbf{R}_0 = \mathbf{D} - \mathbf{B}\mathbf{R}_1$.

2.2 Arithmetic Representation and Big Modulus Representation of Boolean Functions

In order to make the BGG+14 ABE scheme available, Boolean functions need to be expressed as mod$q$ functions, i.e., big modulus functions. This can be easily achieved by firstly transforming each Boolean operation into an arithmetic operation and then transforming the arithmetic operation into a big modulus operation. For example, for two bit variables x_1 and x_2,

$$x_1 \cdot x_2 (\mathrm{mod} 2) = x_1 \cdot x_2 = x_1 \cdot x_2 (\mathrm{mod} q),$$

$$x_1 + x_2 (\mathrm{mod} 2) = x_1 + x_2 - 2x_1 \cdot x_2 = x_1 + x_2 - 2x_1 \cdot x_2 (\mathrm{mod} q).$$

Then, by generalizing these transformations, each operation of a Boolean function can be converted into an operation under a big modulus. Therefore, Boolean functions are described as mod$q$ functions, except that the variables are in $\mathbb{F}_2$ rather than $\mathbb{Z}_q$.

2.3 Quasi-homomorphic Operations of the BGG+14 ABE Scheme

Let $\boldsymbol{x} = (x_1, x_2, \cdots, x_l)$ denote an l-dimensional attribute, where each x_i is a bit variable. Take l matrices $\mathbf{B}_1, \mathbf{B}_2, \cdots, \mathbf{B}_l \in \mathbb{Z}_q^{n\times m}$. Take another l matrices $x_1\mathbf{G} + \mathbf{B}_1, x_2\mathbf{G} + \mathbf{B}_2, \cdots, x_l\mathbf{G} + \mathbf{B}_l \in \mathbb{Z}_q^{n\times m}$. For any *P/poly* Boolean function $f(x)$, there are some 'small-size linear combination operations' for the above matrices, resulting in a new matrix

$$f(x) \cdot \mathbf{G} + \mathbf{B}_f \in \mathbb{Z}_q^{n\times m},$$

where $\mathbf{B}_f$ is independent of x. Recalling Sect. 2.2, any Boolean operation can be viewed as operations in $\mathbb{Z}_q$, and any Boolean function can be viewed as a function in $\mathbb{Z}_q$. Furthermore, for this special function in $\mathbb{Z}_q$, the result of each operation belongs to $[-2, 2]$. We consider the following four simple cases.

Case I. If $f(x) = \alpha x_i$ where α is a constant, then the 'small-size linear combination operation' is

$$(x_i\mathbf{G} + \mathbf{B}_i)\mathbf{G}^{(\alpha)} = \alpha x_i\mathbf{G} + \mathbf{B}_i\mathbf{G}^{(\alpha)}(\mathrm{mod}q),$$

where $\mathbf{B}_f = \mathbf{B}_i\mathbf{G}^{(\alpha)}$.

Case II. If $f(x) = x_i + x_j$, then the 'small-size linear combination operation' is

$$(x_i\mathbf{G} + \mathbf{B}_i) + (x_j\mathbf{G} + \mathbf{B}_j)(\mathrm{mod}q) = (x_i + x_j)\mathbf{G} + (\mathbf{B}_i + \mathbf{B}_j)(\mathrm{mod}q),$$

where $\mathbf{B}_f = \mathbf{B}_i + \mathbf{B}_j$.

Case III. If $f(x) = x_i \cdot x_j$ where $i \leq j$, then the 'small-size linear combination operation' is

$$x_j(x_i\mathbf{G} + \mathbf{B}_i) - (x_j\mathbf{G} + \mathbf{B}_j)\mathbf{G}^{(\mathbf{B}_i)} = x_ix_j\mathbf{G} + (-\mathbf{B}_j\mathbf{G}^{(\mathbf{B}_i)})(\mathrm{mod}q),$$

where $\mathbf{B}_f = -\mathbf{B}_j\mathbf{G}^{(\mathbf{B}_i)}$.

Case IV. If $f(x) = \alpha \cdot x_{j_1} \cdot x_{j_2} \cdot \cdots \cdot x_{j_k}, j_1 \leq j_2 \leq \cdots j_k$ and α is a constant, then the 'small-size linear combination operation' is

$$\sum_{i=1}^{k}\left(\prod_{h=i+1}^{k} x_{j_h}\right) \cdot (x_{j_i}\mathbf{G} + \mathbf{B}_{j_i}) \cdot \mathbf{G}_i = \alpha \cdot x_{j_1} \cdot x_{j_2} \cdot \cdots \cdot x_{j_k} \cdot \mathbf{G} + (-\mathbf{B}_{j_k}\mathbf{G}_k),$$

where $\mathbf{G}_1, \mathbf{G}_2, \cdots, \mathbf{G}_k$ are Boolean matrices in $\mathbb{Z}^{m\times m}$ and are defined recursively as below:

$$\mathbf{G}_1 = \mathbf{G}^{\alpha},$$

$$\mathbf{G}_i = \mathbf{G}^{(-\mathbf{B}_{j_{i-1}}\mathbf{G}_{i-1})}, i = 2, 3, \cdots, k,$$

where $\mathbf{B}_f = -\mathbf{B}_{j_k} \cdot \mathbf{G}$ is also independent of x.

Finally, we affirm that iterations of 'small-size linear combination operations' are still 'small-size linear combination operations', provided the time of iterations

is at the polynomial level. Thus, we draw the conlusion by repeating the aforementioned four operations: any *P/poly* Boolean function f can execute 'small-size linear combination operations' on the above matrices, resulting in $f(x)\mathbf{G} + \mathbf{B}_f$.

Next, we do the following encoding:

$$\mathbf{c}_1 = \mathsf{Encode}(x_1\mathbf{G} + \mathbf{B}_1, \mathbf{s}),$$

$$\mathbf{c}_2 = \mathsf{Encode}(x_2\mathbf{G} + \mathbf{B}_2, \mathbf{s}),$$

$$\cdots,$$

$$\mathbf{c}_l = \mathsf{Encode}(x_l\mathbf{G} + \mathbf{B}_l, \mathbf{s}).$$

By executing the same 'small-size linear combination operation' (only plus a transpose) on the codeword $(\mathbf{c}_1, \mathbf{c}_2, \cdots, \mathbf{c}_l)$, we will obtain

$$\mathbf{c}_f = \mathsf{Encode}(f(x)\mathbf{G} + \mathbf{B}_f, \mathbf{s}).$$

We call such 'small-size linear combination operations' on $(\mathbf{c}_1, \mathbf{c}_2, \cdots, \mathbf{c}_l)$ as quasi-homomorphic operations about the Boolean function f. Here, we need to emphasize two design techniques of the BGG+14 scheme: (i) Case III indicates that when executing the quasi-homomorphic operations of multiplication, the accumulation form of errors is approximately the multiplication of an original error by a binary matrix, rather than the multiplication of two original errors. This decelerates the accumulation of errors to a large extent. (ii) Case III is a particular case of Case IV, whereas Case IV is not repeated operations of Case III. This design technique makes that when executing the quasi-homomorphic operations of continuous multiplication, the accumulation of errors by applying Case IV once is much smaller than by applying Case III multiple times.

When multiplying an original error by a random binary matrix, the size of the resulting error is about $\sqrt{\frac{m}{2}}$ times the size of the original error. Hence, when executing the quasi-homomorphic operations of multiplication, the size of the resulting error is at least about $\sqrt{\frac{m}{2}}$ times the size of one original error, and this statement also holds when executing the quasi-homomorphic operations of continuous multiplication, i.e., Case IV. Continuous multiplications are uncommon for a *P/poly*

function, and even two adjacent operations are both multiplications in a $P/poly$ function, they do not necessarily can be merged into a continuous multiplication. In other words, quasi-homomorphic operations of continuous multiplication, i.e., Case IV, have a limited inhibitory effect on the accumulation of errors. To sum up, it is possible for the BGG+14 scheme that when executing the quasi-homomorphic operations for $P/poly$ functions, the size of the final error reaches superpolynomial. Therefore, the modulus q of the BGG+14 scheme has to be superpolynomially large for processing $P/poly$ functions.

2.4 BGG+14 ABE Scheme[21]

- Generating master key $(\mathbf{mpk}, \mathbf{msk})$: The key generator runs $\mathsf{TrapGen}(n, m, q)$ to obtain $(\mathbf{A}, \mathbf{T})$, then he randomly picks $\mathbf{B}_i \in \mathbf{Z}_q^{n\times m}, i = 1, 2, \cdots, l, \mathbf{D} \in \mathbf{Z}_q^{n\times m}$. The output is

$$\mathbf{mpk} = (\mathbf{A}, \mathbf{B}_1, \cdots, \mathbf{B}_l, \mathbf{D}), \mathbf{msk} = \mathbf{T}.$$

- Generating secret key $\mathbf{sk}_f$ for the Boolean function f: The key generator firstly generates $\mathbf{B}_f$. Note that $\mathbf{B}_f$ is generated by the method in Sect. 2.3. The attribute is randomly chosen, and the resulting $\mathbf{B}_f$ is independent of this attribute. Then, he runs $\mathsf{ReKeyGen}(\mathbf{A}, y_0\mathbf{G} + \mathbf{B}_f, \mathbf{T}, \mathbf{D})$ to obtain $\mathbf{R} \in \mathbb{Z}^{2m\times m}$. The output is

$$\mathbf{sk}_f = \mathbf{R}.$$

- Encryption: The plaintext $\mathbf{m}$ is an m-dimensional Boolean vector. The attribute $\boldsymbol{x} = (x_1, x_2, \cdots,$
  $x_l)$ is sent to the encryptor. The encryptor randomly picks $\mathbf{s} \in \mathbb{Z}_q^n$, and computes $(\mathsf{Encode}(\mathbf{A}, \mathbf{s}),\ \mathsf{Encode}(x_1\mathbf{G} + \mathbf{B}_1, \mathbf{s}),\ \mathsf{Encode}(x_2\mathbf{G} + \mathbf{B}_2, \mathbf{s}), \cdots,\ \mathsf{Encode}(x_l\mathbf{G} + \mathbf{B}_l, \mathbf{s}),\ \mathsf{Encode}(\mathbf{D}, \mathbf{s}))$. The ciphertext is

$$\begin{aligned}\mathbf{C} &= (\mathbf{c}_{in}, \mathbf{c}_1, \mathbf{c}_2, \cdots, \mathbf{c}_l, \mathbf{c}_{out}) \\ &= (\mathsf{Encode}(\mathbf{A}, \mathbf{s}), \mathsf{Encode}(x_1\mathbf{G} + \mathbf{B}_1, \mathbf{s}), \cdots, \mathsf{Encode}(x_l\mathbf{G} + \mathbf{B}_l, \mathbf{s}), \mathsf{Encode}(\mathbf{D}, \mathbf{s}) + \lceil\frac{q}{2}\rceil\mathbf{m}) \\ &= (\mathbf{A}^T\mathbf{s} + \mathbf{e}_{in}, (x_1\mathbf{G} + \mathbf{B}_1)^T\mathbf{s} + \mathbf{e}_1, \cdots, (x_l\mathbf{G} + \mathbf{B}_l)^T\mathbf{s} + \mathbf{e}_l, \mathbf{D}^T\mathbf{s} + \mathbf{e}_{out} + \lceil\frac{q}{2}\rceil\mathbf{m}).\end{aligned}$$

- Decryption: By using his own f and the attribute $\boldsymbol{x} = (x_1, x_2, \cdots, x_l)$, the decryptor executes the quasi-homomorphic operation to obtain

$$\begin{aligned}\mathbf{c}_f &= \mathsf{Encode}(f(\boldsymbol{x})\mathbf{G} + \mathbf{B}_f, \mathbf{s}) \\ &= (f(\boldsymbol{x})\mathbf{G} + \mathbf{B}_f)^T\mathbf{s} + \mathbf{e}_f(\boldsymbol{x}).\end{aligned}$$

Then, by using $\mathbf{sk}_f = \mathbf{R}$, the decryptor computes

$$\begin{aligned}\mathbf{c}_{out} - \mathbf{R}^T \begin{pmatrix} \mathbf{c}_{in} \\ \mathbf{c}_f \end{pmatrix} &= \mathbf{D}^T\mathbf{s} - \mathbf{D}^T\mathbf{s} + \lceil \frac{q}{2} \rceil \mathbf{m} + ((y_0 - f(\boldsymbol{x}))\mathbf{G})^T \mathbf{s} + \mathbf{e}' \\ &= \lceil \frac{q}{2} \rceil \mathbf{m} + ((y_0 - f(\boldsymbol{x}))\mathbf{G})^T\mathbf{s} + \mathbf{e}'.\end{aligned}$$

When $f(\boldsymbol{x}) = y_0$, the plaintext $\mathbf{m}$ can be obtained by using "*Rounding*"; When $f(\boldsymbol{x}) \neq y_0$, gibberish is returned. Generally, $y_0 = 1$.

2.5 Hiding Attribute in the BGG+14 ABE Scheme

The so-called 'hiding attribute' means that the encryptor only knows the attribute and the decryptor does not know it. The key issue is whether the decryptor can execute the quasi-homomorphic operation of f when he does not know some part of the attribute.

It can be easily seen that when f is a mod$q$ linear function, the decryptor can finish the quasi-homomorphic operation without knowing the attribute $\boldsymbol{x}$. However, from Sect. 2.2, any Boolean function is not a mod$q$ linear function. In other words, as a mod$q$ function, any Boolean function contains mod$q$ additions and mod q multiplications. When executing multiplications, i.e., the quasi-homomorphic operation of multiplications, one attribute bit can be secret, while another one has to be visible to the decryptor.

To support the analysis of this article, we extend f and give the following two cases.

Case V. Suppose that the decryptor knows $t'\mathbf{G} + \mathbf{B}'$, $v\mathbf{G} + \mathbf{B}''$, and the bit v, while the decryptor does not know the bit t', $f(t', v) = t' \cdot v \cdot 2^{k'} (\mathrm{mod} p)$, where the modulus $p < q$, then the 'small-size linear combination operation' is

$$v(t'\mathbf{G} + \mathbf{B}') \cdot \mathbf{G}^{(2^{k'} (\bmod p))} - (v\mathbf{G} + \mathbf{B}'') \cdot \mathbf{G}^{(\mathbf{B}'\mathbf{G}^{(2^{k'} (\bmod p))})} = f(t', v) \cdot \mathbf{G} + \mathbf{B}_f,$$

where $\mathbf{B}_f = -\mathbf{B}'' \cdot \mathbf{G}^{(\mathbf{B}'\mathbf{G}^{(2^{k'} \bmod p)})}$ is independent of (t', v). Although the decryptor can execute the quasi-homomorphic operation of $f(t', v)$, he does not know the value of $f(t', v)$. When $v = 0$ he knows that $f(t', v) = 0$; When $v = 1$ he knows that $f(t', v)$ is either zero or $2^{k'} (\mathrm{mod} p)$.

Case VI. Suppose that the decryptor knows $a\mathbf{G} + \mathbf{B}'$ and $b\mathbf{G} + \mathbf{B}''$, but he does not know the two arithmetic values a and b between $(-\frac{p}{2}, \frac{p}{2})$, where $f(a, b) = a + b (\mathrm{mod} p)$ and the modulus $p < q$. Then, there are no 'small-size linear combination operations' of $a\mathbf{G} + \mathbf{B}'$ and $b\mathbf{G} + \mathbf{B}''$, such that the result is $f(a, b) \cdot \mathbf{G} + \mathbf{B}_f$, where $\mathbf{B}_f$ is independent of (a, b). In other words, the extension structures of the BGG+14 scheme do not support the quasi-homomorphic operation of $f(a, b)$. The reason is that $a + b (\mathrm{mod} p)$ has to be converted into a series of mod$q$ operations to complete the quasi-homomorphic operation. It is impossible that these mod$q$ operations only include mod$q$ additions while excluding mod$q$ multiplications. Furthermore, for mod$q$ multiplications, the quasi-homomorphic operation cannot be executed when the values on both sides are unknown.

3 The Agr17 FE Scheme

3.1 Overview of the Agr17 FE Scheme[19]

For a plaintext u, the encryption process can be divided into two steps: (1) u is encrypted to an FHE ciphertext u^* by the encryption algorithm of an FHE scheme; (2) u^* is taken as the public part of the attribute, and t, the secret key of the FHE scheme, is taken as the hidden part of the attribute. Then, for the attribute (u^*, t), a public 'formal plaintext' $\mathbf{m}$ is encrypted to an 'ABE ciphertext' $\mathbf{C}$ by the BGG+14 ABE scheme. Finally, the obtained 'ABE ciphertext' $\mathbf{C}$ is the ciphertext of the Agr17 FE scheme.

Now, the decryptor knows the following four items: (1) The ciphertext $\mathbf{C}$ of the Agr17 FE scheme. In fact, this item is the 'ABE ciphertext' $\mathbf{C} = (\mathbf{c}_{in}, \mathbf{c}_1, \cdots, \mathbf{c}_l, \mathbf{c}_{out})$, where $(\mathbf{c}_1, \cdots, \mathbf{c}_l) = (\mathbf{C}_{u^*}, \mathbf{C}_t)$, $\mathbf{C}_{u^*}$ is the 'ABE ciphertext' of u^* (the public part of the attribute), and $\mathbf{C}_t$ is the 'ABE ciphertext' of t (the hidden part of the attribute);

(2) The FHE ciphertext u^*. In fact, this item is exactly the public part of the attribute; (3) The public 'formal plaintext' $\mathbf{m}$; (4) The secret key of the Agr17 scheme, which is corresponding to the Boolean function f. In fact, this item is the secret key of the BGG+14 scheme, which is corresponding to the composite function Df^* where f^* is the homomorphic operation of f and D is the FHE decryption algorithm. In other words, there is $Df^*(u^*, t) = D(f^*(u^*), t) = f(u)$. The decryptor knows neither the plaintext u nor the secret key of the FHE scheme t. Under such a limitation, the decryptor needs to solve the functional value $f(u)$ of the plaintext u.

The rough decryption process of the decryptor is as follows. He executes the 'functional decryption' (i.e., 'attribute decryption') on the ciphertext $\mathbf{C}$ by using his secret key of the Agr17 scheme, i.e., the secret key of the BGG+14 scheme. If the resulting formal plaintext is $\mathbf{m}$, then $f(u) = 1$, otherwise, $f(u) = 0$.

Note that the ciphertext $\mathbf{C}$ includes two parts $\mathbf{C} = (\mathbf{C}_{u^*}, \mathbf{C}_t)$. Therefore, the detailed decryption process of the decryptor includes the following three steps.

First, $\mathbf{C}_{u^*}$ is transformed into a new ciphertext $\mathbf{C}_{f^*(u^*)}$ by the homomorphic operation of f, where $\mathbf{C}_{f^*(u^*)}$ is the 'ABE ciphertext' of the new attribute $f^*(u^*)$. In other words, this step is the 'quasi-homomorphic operation of the homomorphic operation of f' (cf. Sect. 2.3).

Second, $\mathbf{C}_{f^*(u^*)}$ and $\mathbf{C}_t$ are performed calculations to obtain the final ciphertext $\mathbf{C}_{D(f^*(u^*),t)} = \mathbf{C}_{f(u)}$, where $\mathbf{C}_{f(u)}$ is the 'ABE ciphertext' of the final corresponding attribute $f(u)$. In other words, this step is the 'quasi-homomorphic operation of the homomorphic decryption operation' (cf. Sect. 2.3).

Finally, the decryptor executes the 'attribute decryption' on the final ciphertext $\mathbf{C}_{f(u)}$ by using his secret key of the BGG+14 ABE scheme. If the resulting formal plaintext is $\mathbf{m}$, then the final attribute is $f(u) = 1$, otherwise, $f(u) = 0$.

3.2 Homomorphic Decryption of the Agr17 FE Scheme

Denote the modulus in the early stage of the homomorphic decryption operation as Q. According to the rough description in the Agr17 FE scheme, the homomorphic decryption operation should be the following Algorithm I or Algorithm II.

Algorithm I: Step 1. Executing the inner product of $f^*(u^*)$ and t under the modulus Q. Step 2. Executing mod2.

Algorithm II: Step 1. Executing the inner product of $f^*(u^*)$ and t under the modulus Q. Step 2. Checking whether the result is close to zero or close to $Q/2$. Obtaining zero if it is close to zero and one if it is close to $Q/2$.

We know that Algorithm I is in fact equivalent to Algorithm II. Concretely, multiplying Algorithm II by two can obtain Algorithm I. Nevertheless, neither Algorithm I nor II can be implemented because the decryptor in the Agr17 FE scheme executes the quasi-homomorphic operation of Algorithm I or II, rather than executes Algorithm I or II directly. Since the decryptor does not know t, he cannot obtain the inner product after the quasi-homomorphic operation of Step 1. Instead, he can only obtain the quasi-homomorphic result of the inner product. Under this circumstance, the quasi-homomorphic operation of Step 2 cannot be implemented as Step 2 is not mod$q$ addition (cf. Sect. 2.3 and Sect. 2.5, q is the modulus of the BGG+14 scheme). Based on the modulus reduction technique of FHE, the homomorphic decryption operation of the Agr17 scheme employs an 'outflanking tactic', i.e., the following Algorithm III.

Algorithm III: Step 1. Executing the modulus reduction on $f^*(u^*)$ such that the modulus Q can be scaled down to a modulus p with polynomial size, and hence $f^*(u^*)$ is transformed into a new FHE ciphertext $f^{**}(u^*)$. Step 2. Executing the inner product of $f^{**}(u^*)$ and t under the small modulus p. Step 3. Exhaustion. Multiplying this inner product by each $i \in \{1, 2, \cdots, p-1\}$ under the modulus p. Step 4. For each product, executing the attribute decryption with a secret key of the BGG+14 scheme, and checking whether the formal plaintext is $\mathbf{m}$, i.e., checking whether the product is one. Step 5. So far, the decryptor already knows the result of Step 2. Therefore, the decryptor executes mod 2 on the result of Step 2, and obtains

$f(u)$. Or, the decryptor checks whether the result of Step 2 is close to zero or $p/2$. If it is close to zero, he obtains $f(u) = 0$, otherwise, $f(u) = 1$.

According to our understanding of the Agr17 FE scheme, the quasi-homomorphic operation of each step in Algorithm III can be finished under the BGG+14 scheme without knowing t. The key point is that the original 'mod2 operation' is replaced by the 'modulus reduction and exhaustion'. Of course, there are still several detailed questions remaining to be answered, but they are not essential. For example, why the modulus reduction of t is not necessary? The answer is that t is represented by bit components. Then, why the modulus reduction of $f^*(u^*)$, which can also be represented by bit components, is necessary? This is due to a basic property of the FHE modulus reduction, which is that the modulus reduction of t is not necessary, while that of $f^*(u^*)$ is obligatory. The decryptor knows $f^*(u^*)$, and hence the quasi-homomorphic operation of the modulus reduction on $f^*(u^*)$ can be implemented.

3.3 Multiple Keys of the Agr17 FE Scheme

In order to provide multiple secret keys under the premise of security, the Agr17 scheme extended the BGG+14 scheme. k matrices $\mathbf{D}^{(1)}, \mathbf{D}^{(2)}, \cdots, \mathbf{D}^{(k)} \in \mathbb{Z}_q^{n\times m}$ are randomly chosen, where k is polynomially large, but large enough. All of the decryptors do not know $\{\mathbf{D}^{(1)}, \mathbf{D}^{(2)}, \cdots, \mathbf{D}^{(k)}\}$, yet each decryptor knows his sum matrix $\sum_{i\in\Delta} \mathbf{D}^{(i)} (\mathrm{mod} q)$, where Δ is a subset of $\{1, 2, \cdots, k\}$. Each decryptor has a distinct subset Δ. Each decryptor regards his sum matrix $\sum_{i\in\Delta} \mathbf{D}^{(i)} (\mathrm{mod} q)$ as the matrix $\mathbf{D}$ in the BGG+14 scheme.

In the encryption phase, $\mathbf{c}_{out}$ is extended to the following form:

$$\begin{aligned}\mathbf{c}_{out} = (&\mathsf{Encode}(\mathbf{D}^{(1)}, \mathbf{s}) + \lceil \frac{q}{2} \rceil \mathbf{m}^{(1)},\\ &\mathsf{Encode}(\mathbf{D}^{(2)}, \mathbf{s}) + \lceil \frac{q}{2} \rceil \mathbf{m}^{(2)},\\ &\cdots,\\ &\mathsf{Encode}(\mathbf{D}^{(k)}, \mathbf{s}) + \lceil \frac{q}{2} \rceil \mathbf{m}^{(k)}).\end{aligned}$$

The encryptor publishes $\{\mathbf{m}^{(1)}, \mathbf{m}^{(2)}, \cdots, \mathbf{m}^{(k)}\}$. When executing decryption, the decryptor who knows $\mathbf{D} = \sum_{i\in\Delta} \mathbf{D}^{(i)}$ and Δ knows $\mathbf{m} = \sum_{i\in\Delta} \mathbf{m}^{(i)} (\mathrm{mod} 2)$, and he

knows

$$\begin{aligned}
&\sum_{i\in\Delta}(\mathsf{Encode}(\mathbf{D}^{(i)},\mathbf{s})+\lceil\frac{q}{2}\rceil\mathbf{m}^{(i)})\\
=&\mathsf{Encode}\left(\sum_{i\in\Delta}\mathbf{D}^{(i)},\mathbf{s}\right)+\lceil\frac{q}{2}\rceil\left(\sum_{i\in\Delta}\mathbf{m}^{(i)}(\mathrm{mod}2)\right)\\
=&\mathsf{Encode}(\mathbf{D},\mathbf{s})+\lceil\frac{q}{2}\rceil\mathbf{m}.
\end{aligned}$$

4 Invalidity of the Agr17 FE Scheme

4.1 Our Finding: Double Modulus Reduction

Now, we focus on the homomorphic decryption operation of the Agr17 FE scheme, i.e., Algorithm III in Sect. 3.2. Step 1 is the modulus reduction. Although it has been described in the Agr17 scheme [19] and elaborated in Sect. 3 in this paper, we find a problem: there are two moduli before the modulus reduction, one is the modulus Q at the early stage of FHE, and the other is the modulus q at the early stage of the BGG+14 ABE scheme. We call Q and q as an inner modulus and outer modulus, which are used for FHE and the quasi-homomorphic operations of FHE, respectively. Operations under the inner modulus can be represented as a series of Boolean operations, and hence the quasi-homomorphic operations can be implemented, which can be represented by operations under the outer modulus. The modulus reduction stated in the Agr17 scheme should be interpreted as the inner modulus reduction. In addition, the inner modulus reduction should be implemented by a series of operations under the outer modulus, i.e., the inner modulus reduction should be transformed into quasi-homomorphic operations. Therefore, the outer modulus keeps unchanged during the inner modulus reduction.

Next, Step 2 of Algorithm III is conducted, which is the inner product of t and $f^{**}(u^*)$ under the small modulus p. Note that t and $f^{**}(u^*)$ are represented as bit components, each of which has an encoding. Thus, Step 2 of Algorithm III can be divided into the following two half steps. Step 2.1. Multiplying each bit of t, the corresponding bit of $f^{**}(u^*)$ and some power of two under the modulus p, i.e., Case V in Sect. 2.5. Step 2.2. Executing addition under the modulus p on these mod$p$ products, i.e., Case VI in Sect. 2.5. Since the quasi-homomorphic operations are

infeasible for Case VI, Step 2 of Algorithm III cannot be finished. Of course, this step is viable if $p = q$, and this is supported by the BGG+14 scheme (cf. Sect. 2.5).

Another serious obstacle is that when f ia a *P/poly* function, f^* is also a *P/poly* function, and hence Q and q are both superpolynomially large (cf. the last paragraph in Sect. 2.3). This means that the outer modulus q cannot be equal to the small modulus p, which is polynomially large. Instead, another modulus reduction is needed. Then, whether the inner modulus Q and outer modulus q can be reduced to the small modulus p with a polymonial size in a single step? It can be seen from the structure of the BGG+14 scheme that this is impossible.

Based on the above analysis, Step 1 of Algorithm III, i.e., the modulus reduction, must be divided into two steps: the inner modulus reduction $Q \to p$ and the outer modulus reduction $q \to p$.

4.2 The Inner Modulus Reduction: $Q \to p$

The inner modulus reduction is simple, which can be viewed as a series of Boolean operations. The inner modulus Q is reduced to the small modulus p, and meanwhile the homomorphic function $f^*(u^*)$ is replaced by a new homomorphic function $f^{**}(u^*)$. We have the following notes for the inner modulus reduction.

Note I: Obviously, its actual operation is a series of quasi-homomorphic operations of Boolean operations, which transform the 'ABE ciphertext' $\mathbf{C}_{f^*(u^*)}$ into a new ABE ciphertext $\mathbf{C}_{f^{**}(u^*)}$. Specifically, before the inner modulus reduction, there is

$$\mathbf{C}_{f^*(u^*)} = \begin{pmatrix} \mathbf{c}_{f_1^*(u^*)} \\ \mathbf{c}_{f_2^*(u^*)} \\ \vdots \\ \mathbf{c}_{f_{k^*}^*(u^*)} \end{pmatrix} = \begin{pmatrix} \mathsf{Encode}(f_1^*(u^*)\mathbf{G} + \mathbf{B}_{f_1^*}, \mathbf{s}) \\ \mathsf{Encode}(f_2^*(u^*)\mathbf{G} + \mathbf{B}_{f_2^*}, \mathbf{s}) \\ \vdots \\ \mathsf{Encode}(f_{k^*}^*(u^*)\mathbf{G} + \mathbf{B}_{f_{k^*}^*}, \mathbf{s}) \end{pmatrix}.$$

After the inner modulus reduction, there is

$$\mathbf{C}_{f^{**}(u^*)} = \begin{pmatrix} \mathbf{c}_{f_1^{**}(u^*)} \\ \mathbf{c}_{f_2^{**}(u^*)} \\ \vdots \\ \mathbf{c}_{f_{k^{**}}^{**}(u^*)} \end{pmatrix} = \begin{pmatrix} \mathsf{Encode}(f_1^{**}(u^*)\mathbf{G} + \mathbf{B}_{f_1^{**}}, \mathbf{s}) \\ \mathsf{Encode}(f_2^{**}(u^*)\mathbf{G} + \mathbf{B}_{f_2^{**}}, \mathbf{s}) \\ \vdots \\ \mathsf{Encode}(f_{k^{**}}^{**}(u^*)\mathbf{G} + \mathbf{B}_{f_{k^{**}}^{**}}, \mathbf{s}) \end{pmatrix},$$

where $f^{**}(u^*)$ is another homomorphic operation of $f(u)$. Different from $f^*(u^*)$, $f^{**}(u^*)$ is the message under the small modulus p, and satisfies the mod p decryption equation

$$(< f^{**}(u^*), t > \mathrm{mod} p) \bmod 2 = f(u) \text{ or } \mathit{Rounding}(< f^{**}(u^*), t > \mathrm{mod} p).$$

Of course, in the actual mod$p$ decryption phase, mod2 or *Rounding* cannot be executed after finishing $< f^{**}(u^*), t >$ mod p. Instead, the decryption process executes exhaustion and the attribute decryption.

Note II: Due to Note I, the outer modulus should keep unchanged during the inner modulus reduction.

4.3 Invalidity of the Outer Modulus Reduction $q \to p$

The ciphertext $\mathbf{C} = (\mathbf{c}_{in}, \mathbf{C}_{u^*}, \mathbf{C}_t, \mathbf{c}_{out})$ is transformed into $(\mathbf{c}_{in}, \mathbf{C}_{f^*(u^*)}, \mathbf{C}_t, \mathbf{c}_{out})$ by the quasi-homomorphic operations of homomorphic operations, and then into $(\mathbf{c}_{in}, \mathbf{C}_{f^{**}(u^*)}, \mathbf{C}_t, \mathbf{c}_{out})$ by the quasi-homomorphic operations of the inner modulus reduction. Next, the outer modulus reduction is conducted.

Complied with the structure of the BGG+14 scheme, the outer modulus reduction is an LWE modulus reduction. The so-called LWE modulus reduction refers to the following two steps: (1) Executing the operation $\frac{p}{q} \times (\cdot)(\bmod p)$ on the ciphertext; (2) Executing the operation $\frac{p}{q} \times (\cdot)(\mathrm{mod} p)$ on the matrix used for encoding. Only when these two steps are completed, can the subsequent LWE decryption operation be completed. Because the decryptor does not know the FHE secret key t, he does not know the matrix used for encoding in the ciphertext $\mathbf{C}_t$ and cannot execute the outer modulus reduction for $\mathbf{C}_t$. It is required that $\mathbf{C}_t$ is constructed to a message under the small modulus p in the encryption phase. Besides, in order to make the Agr17 scheme proceed as smoothly as possible, we suppose that $(\mathbf{c}_{in}, \mathbf{C}_t, \mathbf{c}_{out})$ are all messages under the small modulus p, rather than the modulus q. This means the trapdoor $\mathbf{T}$ is also constructed according to the small modulus p.

Now, the outer modulus reduction is only conducted for $\mathbf{C}_{f^{**}(u^*)}$:

$$(1)\frac{p}{q} \times \mathbf{C}_{f^{**}(u^*)}(\mathrm{mod} p);$$

$$(2)\frac{p}{q} \times \begin{pmatrix} f_1^{**}(u^*)\mathbf{G} + \mathbf{B}_{f_1^{**}} \\ f_2^{**}(u^*)\mathbf{G} + \mathbf{B}_{f_2^{**}} \\ \vdots \\ f_{k^{**}}^{**}(u^*)\mathbf{G} + \mathbf{B}_{f_{k^{**}}^{**}} \end{pmatrix} (\mathrm{mod} p).$$

These two steps can be completed, but the subsequent decryption would not be executed. The reason lies in that the decryption of the BGG+14 scheme is an LWE decryption with conditions rather than an ordinary LWE decryption, and the outer modulus reduction will destroy the conditions of decryption.

5 Two Revised Versions and Their Weaknesses

5.1 First Revised Version and Efficiency

First, suppose that $(\mathbf{c}_{in}, \mathbf{c}_t, \mathbf{c}_{u^*}, \mathbf{c}_{out})$ are messages under the big modulus q, rather than the small modulus p. Therefore, the trapdoor $\mathbf{T}$ is still constructed according to the big modulus q. Then, our first revised version keeps the original Agr17 scheme for the front part until the inner modulus reduction is finished and the corresponding ABE ciphertext of $f^{**}(u^*)$ is obtained. Then, we consider the arithmetic inner product $< t, f^{**}(u^*) >$ of the FHE secret key t and the FHE ciphertext $f^{**}(u^*)$, i.e., the inner product without the modulus, rather than the inner product under the small modulus p: $< t, f^{**}(u^*) > (\mathrm{mod} p)$. Note that this arithmetic inner product is much larger than the small modulus p, but is still polynomially large. Therefore, this arithmetic inner product can be viewed as the inner product under the big modulus q: $< t, f^{**}(u^*) > = < t, f^{**}(u^*) > (\mathrm{mod} q)$, and then the quasi-homomorphic operations can be implemented.

Then, guessing possible results of this arithmetic inner product, which is polynomially many. For each non-zero possible result a, multiplying $a^{-1}(\mathrm{mod} q)$ by the corresponding ABE ciphertext $< t, f^{**}(u^*) > (\mathrm{mod} q)$. The quasi-homomorphic operations of this step are viable.

Next, for each non-zero possible result a, considering the function $a^{-1} < t, f^{**}(u^*) > (\mathrm{mod} q)$ of the attribute (t, u^*), asking for the ABE decryption key for this funtion and conducting the decryption.

So far, the decryptor knows exactly the arithmetic inner product $< t, f^{**}(u^*) >$, for which he executes mod$p$ and then mod2 (or '*Rounding*'). In this way, the FE decryption is finished, indicating that the first revised version is effective.

Efficiency: In the first revised version, the number of the decryption keys is much larger than the small modulus p, which is the number of the decryption keys in the original Agr17 scheme. This means the efficiency of the first revised version is considerably lower than the original Agr17 scheme.

5.2 Insecurity of the First Revised Version

The first revised version is a 'natural' modification of the Agr17 scheme. In the first revised version, the decryptor knows the arithmetic inner product $< t, f^{**}(u^*) >$. Nevertheless, it is only required that the decryptor knows $< t, f^{**}(u^*) >$ (mod$p$) in the original Agr17 scheme. Consequently, the first revised version leaks more information about the secret key t.

Note that t and $f^{**}(u^*)$ are both represented by bit components, while the arithmetic inner product $< t, f^{**}(u^*) >$ is actually the sum of the product of each bit of t, the corresponding bit of $f^{**}(u^*)$, and the corresponding power of two. Therefore, Boolean linear equations of t can be obtained by executing mod 2 on the arithmetic inner product. Then, the decryptors can solve t by collusion attacks with much less cost and break the first revised scheme.

5.3 Second Revised Version and Its Limitations

Our second revised version is an application of verification circuit technology of GGH+13b [17]. A verification circuit f' of the Boolean function f takes several middle variables of f as a part of its 'independent variables'. That is, $f'(u_1, \cdots, u_l, v_1, \cdots, v_k) = f(u_1, \cdots, u_l)$, where all 'independent variables' of f' are all independent variables, $\{u_1, \cdots, u_l\}$, and several middle variables of f, $\{v_1, \cdots, v_k\}$. Suppose f is a $P/poly$ function, while f' is an NC^1 circuit. The technical routine is as follows.

(1) Encrypt $\{u_1, \cdots, u_l, v_1, \cdots, v_k\}$ by using FHE to obtain $\{u_1^*, \cdots, u_l^*, v_1^*, \cdots, v_k^*\}$.

(2) Take $\{u_1^*, \cdots, u_l^*, v_1^*, \cdots, v_k^*\}$ as one part of the attribute, the FHE decryption key t as another part of the attribute, run the BGG+14 ABE encryption.

(3) Run the quasi-homomorphic operation of $f'^*(u_1^*, \cdots, u_l^*, v_1^*, \cdots, v_k^*)$, where f'^* is the FHE homomorphic evaluation of $f'(u_1, \cdots, u_l, v_1, \cdots, v_k)$. (Note: $f' \in NC^1 \rightarrow f'^* \in NC^1 \rightarrow$ both f'^* and the quasi-homomorphic operation of f'^* gather polynomially large errors $\rightarrow$ both of the FHE modulus Q and the BGG+14 ABE modulus q can be taken as polynomially large).

(4) Modulus switching. Run the quasi-homomorphic operation of such operation: change the text f'^* under the modulus Q into a text f'^{**} under the modulus q. (Note: we do not know how to set $Q = q$ at the beginning, and $Q < q$ is possible. Then the modulus switching makes $Q = q$).

(5) Run the quasi-homomorphic operation of the modulus inner product $< t, f'^{**} > (\mathrm{mod} q)$, where $f'^{**}(u_1^* \sim u_l^*, v_1^* \sim v_k^*)$ is known while t is unknown.

(6) Search the value of $< t, f'^{**} > (\mod q)$ by using q decryption keys of the BGG+14 ABE scheme. Then executing mod2 (or *Rounding*). Then the FE decryption is completed, and the value of $f'(u_1 \sim u_l, v_1 \sim v_k) = f(u_1 \sim u_l)$ is obtained.

The second revised version is far from the design idea of the Agr17 scheme. On the other hand, the second revised version greatly limits the function class. It only allows the functions which have NC^1 circuits when 'independent variables' expands from $(u_1 \sim u_l)$ to special $(u_1 \sim u_l, v_1 \sim v_k)$, rather than any *P/poly* function.

References

1. Boneh, D., Sahai, A., Waters, B.: Functional encryption: definitions and challenges. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 253–273. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-19571-6_16
2. Boneh, D., Franklin, M.: Identity-based encryption from the weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44647-8_13
3. Cocks, C.: An identity based encryption scheme based on quadratic residues. In: Honary, B. (ed.) Cryptography and Coding 2001. LNCS, vol. 2260, pp. 360–363. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-45325-3_32
4. Boyen, X., Waters, B.: Anonymous hierarchical identity-based encryption (without random oracles). In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 290–307. Springer, Heidelberg (2006). https://doi.org/10.1007/11818175_17

5. Boneh, D., Waters, B.: Conjunctive, subset, and range queries on encrypted data. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 535–554. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-70936-7_29
6. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: STOC 2008. pp. 197–206. ACM (2008) https://doi.org/10.1145/1374376.1374407
7. Cash, D., Hofheinz, D., Kiltz, E., Peikert, C.: Bonsai trees, or how to delegate a lattice basis. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 523–552. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13190-5_27
8. Agrawal, S., Boneh, D., Boyen, X.: Efficient lattice (H)IBE in the standard model. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 553–572. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13190-5_28
9. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: CCS 2006. pp. 89–98. ACM (2006). https://doi.org/10.1145/1180405.1180418
10. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: IEEE Symposium on Security and Privacy 2007. pp. 321–334. IEEE (2007). https://doi.org/10.1109/SP.2007.11
11. Katz, J., Sahai, A., Waters, B.: Predicate encryption supporting disjunctions, polynomial equations, and inner products. In: Smart, N. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 146–162. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78967-3_9
12. Lewko, A., Okamoto, T., Sahai, A., Takashima, K., Waters, B.: Fully secure functional encryption: attribute-based encryption and (hierarchical) inner product encryption. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 62–91. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13190-5_4
13. Agrawal, S., Freeman, D.M., Vaikuntanathan, V.: Functional encryption for inner product predicates from learning with errors. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 21–40. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25385-0_2
14. Waters, B.: Functional encryption for regular languages. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 218–235. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32009-5_14
15. 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
16. Garg, S., Gentry, C., Halevi, S., Sahai, A., Waters, B.: Attribute-based encryption for circuits from multilinear maps. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8043, pp. 479–499. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40084-1_27
17. Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. In: FOCS 2013. pp. 40–49. IEEE (2013). https://doi.org/10.1109/FOCS.2013.13
18. Gorbunov, S., Vaikuntanathan, V., Wee, H.: Predicate encryption for circuits from LWE. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015. LNCS, vol. 9216, pp. 503–523. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48000-7_25

19. 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

20. Lai, Q., Liu, FH., Wang, Z.: New lattice two-stage sampling technique and its applications to functional encryption – stronger security and smaller ciphertexts. In: Canteaut, A., Standaert, FX. (eds.) EUROCRYPT 2021. LNCS, vol. 12696, pp. 498–527. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-77870-5_18

21. Boneh, D., Gentry, C., Gorbunov, S., Halevi, S., Nikolaenko, V., Segev, G.,Vaikuntanathan, V., and Vinayagamurthy, D.: Fully key-homomorphic encryption, arithmetic circuit ABE, and compact garbled circuits. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 533–556. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-55220-5_30

22. Wang, Z., Fan X., Liu FH.: FE for inner products and its application to decentralized ABE. In: Lin, D., Sako, K. (eds.) PKC 2019. LNCS, vol. 11443, pp. 97–127. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-17259-6_4

23. Ananth, P., Vaikuntanathan, V.: Optimal bounded-collusion secure functional encryption. In: Hofheinz, D., Rosen, A. (eds.) TCC 2019. LNCS, vol. 11891, pp. 174–198. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-36030-6_8

24. Genise, N., Micciancio, D.: Faster gaussian sampling for trapdoor lattices with arbitrary modulus. In: Nielsen, J., Rijmen, V. (eds.) EUROCRYPT 2018. LNCS, vol. 10820, pp. 174–203. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-78381-9_7

25. Agrawal, S.: Indistinguishability obfuscation without multilinear maps: new methods for bootstrapping and instantiation. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019. LNCS, vol. 11476, pp. 191–225. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-17653-2_7

26. Agrawal, S., Rosen, A.: Functional encryption for bounded collusions, revisited. In: Kalai, Y., Reyzin, L. (eds.) TCC 2017. LNCS, vol. 10677, pp. 173–205. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70500-2_7

27. Datta, P., Okamoto, T., Takashima, K.: Adaptively simulation-secure attribute-hiding predicate encryption. In: Peyrin, T., Galbraith, S. (eds.) ASIACRYPT 2018. LNCS, vol. 11273, pp. 640–672. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-03329-3_22

28. Agrawal, S., Yamada, S.: Optimal broadcast encryption from pairings and LWE. In: Canteaut, A., Ishai, Y.: (eds.) EUROCRYPT 2020. LNCS, vol. 12105, pp. 13–43. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45721-1_2

29. Agrawal, S., Maitra, M., Yamada, S.: Attribute based encryption (and more) for nondeterministic finite automata from LWE. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019. LNCS, vol. 11693, pp. 765–797. Springer, Cham (2019) https://doi.org/10.1007/978-3-030-26951-7_26

30. Chen, Y., Vaikuntanathan, V., Waters, B., Wee, H., Wichs, D.: Traitor-tracing from LWE made simple and attribute-based. In: Beimel, A., Dziembowski, S. (eds.) TCC 2018. LNCS, vol. 11240, pp. 341–369. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-03810-6_13

31. Agrawal, S., Libert, B., Maitra, M., Titiu, R.: Adaptive simulation security for inner product functional encryption. In: Kiayias, A., Kohlweiss, M., Wallden, P., Zikas, V. (eds.) PKC 2020. LNCS, vol. 12110, pp. 34–64. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45374-9_2
32. 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. LNCS, vol. 11694, pp. 284–332. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-26954-8_10
33. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (Leveled) fully homomorphic encryption without bootstrapping. In: ITCS 2012. pp. 309–325. ACM (2012). https://doi.org/10.1145/2090236.2090262
34. Gentry, C., Sahai, A., Waters, B.: Homomorphic encryption from learning with errors: conceptually-simpler, asymptotically-faster, attribute-based. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8042, pp. 75–92. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40041-4_5
35. Brakerski, Z., Vaikuntanathan, V.: Lattice-based FHE as secure as PKE. In: ITCS 2014. pp. 1–12. ACM (2014). https://doi.org/10.1145/2554797.2554799