entropy

MDPI

Article

Two-Party Privacy-Preserving Set Intersection with FHE

Yunlu Cai [1], Chunming Tang [1,2,*] and Qiuxia Xu [3]

1 School of Mathematics and Information Science, Guangzhou University, Guangzhou 510006, China; caiyunlu@gzhu.edu.cn
2 State Key Laboratory of Cryptology, P.O. Box 5159, Beijing 100878, China
3 School of Mathematics and Systems Science, Guangdong Polytechnic Normal University, Guangzhou 510665, China; xia_mi0622@126.com
* Correspondence: ctang@gzhu.edu.cn

Received: 28 October 2020; Accepted: 22 November 2020; Published: 25 November 2020

Abstract: A two-party private set intersection allows two parties, the client and the server, to compute an intersection over their private sets, without revealing any information beyond the intersecting elements. We present a novel private set intersection protocol based on Shuhong Gao's fully homomorphic encryption scheme and prove the security of the protocol in the semi-honest model. We also present a variant of the protocol which is a completely novel construction for computing the intersection based on Bloom filter and fully homomorphic encryption, and the protocol's complexity is independent of the set size of the client. The security of the protocols relies on the learning with errors and ring learning with error problems. Furthermore, in the cloud with malicious adversaries, the computation of the private set intersection can be outsourced to the cloud service provider without revealing any private information.

Keywords: private set intersection; privacy-preserving; fully homomorphic encryption; secure multiparty computation

1. Introduction

In 1978, Rivest first presented the idea of fully homomorphic encryption (FHE) [1]. Gentry constructed the first specific FHE scheme in 2009 [2]. Since then, dramatic progress in FHE is made by Gentry and many other researchers around the world. The first generation is based on an approximate GCD problem of integers and ideal lattices [2,3]; the second generation is based on ring learning with errors (RLWE) and learning with errors (LWE) problems, and developed several techniques, including re-linearization, key switch and modulus reduction, for decreasing noise growth [4,5]; the third generation involves the GSW scheme, which is based on approximate eigenvalues and RLWE [6]. Shuhong Gao's scheme [7] is a compressed fully homomorphic encryption scheme, denoted by SGFHE below, and this scheme has three features: (1) The cipher with private key encryption is expanded six times and with public key encryption is $10 + log2(n)$, where n (a power of 2) is the block length of the message; the computation of all ciphertexts is modulo r, where $r = 16n$; and the boundary of noise size is $n - 1$. (2) The bootstrapping algorithm needs only a bootstrapping key and the boundaries of the noise size of the output ciphers are still $n - 1$ with no failure at all. (3) the security of Shuhong Gao's scheme is based on the learning with errors problems and ring learning with errors problems, and for the block length of any message $n \geq 512$, it costs at least 2^{160} bit operations for breaking the scheme with the current approaches. In addition, with TFHE bootstrapping [8], the LWE cipher produced could be invalid with a probability of about 2^{-33} (for $n = 500$). That probability is very small, and for computing many functions it is useful; however, it cannot be applied to functions

that require more than 2^{33} bit operations (unless increasing n). In SGFHE, the error size of the LWE ciphers after bootstrapping are always bound by $n - 1$; this feature is not available in other FHE schemes. The total time cost for the bootstrapping procedure of the SGFHE scheme is about 130 ms, that is, 10 times as much as TFHE.

Secure multi-party computing (SMPC) is mainly about how to compute a function safely without a trusted third party. Secure multi-party computing was first proposed by Yao Qizhi in 1982. After being developed by Goldreich, Micali, Wigderson et al. [9], secure multi-party computing became a very active research field in modern cryptography. The research on MPC [10] is divided into general schemes and specific schemes designed for certain computing scenarios; the general scheme is not as efficient as a specific optimized scheme that is specially designed for a certain application. In practical applications, specific schemes are more widely used [11]. Secret sharing [12], garbled circuit [13,14], oblivious transfer [15], commitment schemes [16] and homomorphic encryption [17] are the key pieces of technology to realize SMPC, and SMPC is of great significance in the study of secret sharing schemes and privacy protection, where it is widely used in correlation analysis, data security queries, trusted data exchanges, etc. [18–22].

Private preserving set intersection (PSI) computing is an important aspect in secure multi-party computing. It not only performs well in scientific computing, but in real life many data can be represented by sets, so it can be used in privacy protection computing to complete corresponding data computing in the sets. The private preserving set intersection computing is the basic operation in many applications, such as machine learning, data mining [23], secure distributed data connection [24] and in privacy protection law enforcement, where it is especially widely used.

1.1. Related Work

Several specialized PSI protocols have been proposed in the literature which are more efficient than using general secure computation [33]. The main methods are: based on oblivious polynomial evaluation [25], based on an oblivious pseudo-random function [26], based on a blind signature [27], based on homomorphic encryption [28], based on the Bloom filter [29], etc. Shen Liyan et al. [30] gave a detailed overview of the development prospects of private preserving set intersection computing, the protocol developed by Google scholar. Mihaela Ion et al. [11] applied private preserving set intersection computing to advertising cooperation.

1.2. Contributions

We present three private set intersection protocols. First, we propose a novel private set intersection protocol based on Shuhong Gao's fully homomorphic encryption scheme and prove the security of the protocol in the honest-but-curious model. We then present a variant of promoted protocol. We also present a variant of the protocol which is a completely novel construction for computing the intersection based on the Bloom filter and a fully homomorphic encryption; this protocol's complexity is independent of the set size of the client. The security of the protocol relies on the learning with errors and ring learning with errors problems. Furthermore, in a cloud with malicious adversaries, the computation of the private set intersection can be outsourced to the cloud service provider without revealing any private information. The ciphertext extension of the protocols is small so that the protocols have strong practicability.

The remainder of the paper is structured as follows: We next review the basic concepts and techniques used in Section 2. In Section 3, we introduce the homomorphic operation used. We describe the basic two-party computing protocol, the improvement protocol and the two-party computing protocol based on the Bloom filter in Section 4. We present our conclusions in Section 5.

2. Basic Concepts and Techniques

2.1. Notation

Let χ be an error distribution; according to the distribution χ, $x \leftarrow \chi$ is randomly chosen. For an integer $n \geq 1$, let $R_n = \mathbb{Z}[x]/(x^n+1)$, $R_{n,q} = \mathbb{Z}[x]/(x^n+1,q)$, where (x^n+1,q) represents the ideal of $\mathbb{Z}[x]$ generated by x^n+1 and q. For any polynomial $f(x) = \sum_{i=0}^{d} f_i(x^i) \in \mathbb{R}(x)$, we define the ∞-norm as $||f(x)||_\infty = \max\limits_{0\leq i\leq d} |f_i|$.

2.2. LWE Ciphers and Modulus Reduction

Regev proposed LWE problem [31,32] over $\mathbb{Z}_q$. Let χ be a probabilistic distribution, and $s \in \mathbb{Z}_q^n$ be an arbitrary vector that is a secret key of any user. $(\mathbf{a}, b)$ is an LWE sample, where $\mathbf{a} \in \mathbb{Z}_q^n$ is selected randomly and uniformly, $b \equiv \langle s, \mathbf{a}\rangle + e \pmod q$, $e \leftarrow \chi$.

Let $D_q = \lfloor q/4 \rfloor$, $1 \leq \tau \leq D_q/2$, $\mathbf{a} \leftarrow \mathbb{Z}_q^n$, and compute $b \equiv \langle \mathbf{s}, \mathbf{a}\rangle + e + xD_q \pmod q$ for encrypting one bit, $e \in [-\tau, \tau]$. Let $E_s(x) = (\mathbf{a}, b) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$, $(\mathbf{a}, b)$ is the LWE ciphertext for $x \in \{0,1\}$. Note that $D_q = \lfloor q/2 \rfloor$ in Regev's but $D_q = \lfloor q/4 \rfloor$ in SGFHE scheme for homomorphic bit operations.

Modulus reduction can reduce the LWE ciphers of $\mathbb{Z}_q$ to $\mathbb{Z}_r$ where r is far less than q.

Lemma 1 ([7]). *Let $\boldsymbol{s}, \boldsymbol{a} \in \mathbb{Z}_q^n$, $e \in \mathbb{Z}_q^n$ with $|e| \leq \tau$, $D_r = \lfloor r/4 \rfloor$, and $b \equiv \langle \boldsymbol{s}, \boldsymbol{a}\rangle + e + xD_q \pmod q$.*

(1) Suppose $\tau \in q(n-3)/(2r), q \geq 4r$ and $\boldsymbol{s} \in \{0,1\}^n$. $b' = \lfloor rb/q \rceil$, $\boldsymbol{a}' = \lfloor r\boldsymbol{a}/q \rceil$; then

$$b' \equiv \langle \boldsymbol{s}, \boldsymbol{a}' \rangle + e + xD_r \pmod r.$$

(2) Let $\ell = \left\lceil log_2^q \right\rceil$, $q \geq 16$. Suppose that $\tau \leq q(n\ell - 5)/(2r)$ with $s \leftarrow \mathbb{Z}_q^n$. Then there exist $\boldsymbol{s}' \in \{0,1\}^{nl}$, $\boldsymbol{a}' \in \mathbb{Z}_r^{n\ell}$ and $b' \in \mathbb{Z}_r$, satisfying

$$b' \equiv \boldsymbol{s}'(\boldsymbol{a}')^t + e' + xD_r \pmod r,$$

where $e' \in \mathbb{Z}$, $|e'| \leq n\ell$.

2.3. RLWE Ciphers

Lyubashevsky et al. introduced the RLWE problem to acquire more efficient encryption schemes [33]. An RLWE sample $\mathbf{v} = (a(x), b(x)) \in \mathbb{R}_n^2$, where $a(x) \leftarrow R_{n,q}$, $a(x) = \sum_{i=0}^{n-1} a_i x^i$, and

$$b(x) := s(x)a(x) + e(x) \pmod{(x^n+1,q)}$$

for some $e(x) \leftarrow R_n$, $||e(x)||_\infty \leq \tau$, τ is the bound of error.

$$\mathbf{v}(-s(x), 1)^t \equiv e(x) \pmod{(x^n+1,q)}.$$

Let $m(x) = \sum_{i=0}^{n-1} m_i x^i$, where $m_i \in \{0,1\}$ denotes an n-bit message. The RLWE cipher of $m(x)$ with error size τ is

$$RE_s(m(x)) = \mathbf{v} + m(x)D_q(0,1) \in R_{n,q}^2.$$

Suppose $RE_s(m(x)) = (a(x), b(x))$. We have

$$b(x) - s(x)a(x) \equiv m(x)D_q + e(x) \pmod{(x^n+1,q)},$$

when $\tau \leq D_q/2$, the message $m(x)$ can be recovered from $m(x) \equiv b(x) - s(x)a(x) \pmod{(x^n+1,q)}$.

2.4. GSW Ciphers and External Product

2.4.1. Gadget Matrix

Suppose that B and l are positive integers so that $B^\ell \geq q$. Suppose that when $g = (1, B, \cdots, B^{\ell-1})$, an arbitrary $a \in \mathbb{Z}_q$ could be denoted by

$$a = a_0 + a_1 + \cdots + a_{\ell-1}B^{\ell-1} = (a_0 + a_1, \cdots + a_{\ell-1})g^t,$$

where $a_i \in \mathbb{Z}$ has a small size. Let $-B/2 \leq a_i \leq B/2$; then $(a_0 + a_1, \cdots + a_{\ell-1})$ is unique. Let $-2B \leq a_i \leq 2B$; the lemma as following is straightforward to prove.

Lemma 2 ([7]). *Let $B^\ell \geq q$, $a \in \mathbb{Z}$. For $0 \leq i \leq \ell - 1$, choose $x_i \leftarrow \mathbb{Z}, |x_i| \leq 3B/2$, which is uniform, random and independent. Suppose that*

$$\begin{aligned} &a - (x_0 + x_1B + \cdots + x_{\ell-1}B^{\ell-1}) \\ &\equiv y_0 + y_1B + \cdots + y_{\ell-1}B^{\ell-1} \pmod{q} \end{aligned}$$

where $|y_i| \leq B/2$. Set $a_i = x_i + y_i$; then $(a_0, a_1, \cdots, a_{\ell-1})$ is uniform random solution to

$$a \equiv a_0 + a_1B + \cdots + a_{\ell-1}B^{\ell-1} \pmod{q}$$

with $|a_i| \leq B/2$.

Hence, any list of elements in $\mathbb{Z}_q$ can be extended. That is, each polynomial $a(x) \in R_{n,q}$ can be denoted by

$$\begin{aligned} a(x) &= a_0(x) + a_1(x)B + \cdots + a_{\ell-1}(x)B^{\ell-1} \\ &= (a_0(x), a_1(x), \cdots, a_{\ell-1}(x))g^t, \end{aligned}$$

where $||a_i(x)||_\infty \leq 2B$. A gadget matrix of $(2\ell) \times 2$ is defined as

$$G = \begin{pmatrix} g^t & 0 \\ 0 & g^t \end{pmatrix}.$$

Any $(a(x), b(x)) \in R_{n,q}^2$ can be denoted by

$$(a(x), b(x)) = \mathbf{u}(x)G \tag{1}$$

where $\mathbf{u}(x) \in R_n^{2\ell}$ is selected randomly and uniformly, and $||\mathbf{u}(x)||_\infty \leq 2B$. Here G^{-1}, only as an operator, acts on the right of $(a(x), b(x))$(G is not a square matrix, so it has no inverse).

$$\mathbf{u}(x) = (a(x), b(x)) \triangleleft G^{-1}$$

A row vector $\mathbf{u}(x)$ has 2ℓ polynomials; the coefficients of the polynomials are small and at most $2B$. This can increase the dimension to decease the coefficient. By the above definition, we have the following equation.

$$(\mathbf{v} \triangleleft G^{-1})G = \mathbf{v}, \forall \mathbf{v} \in R_{n,q}^2$$

2.4.2. External Product

Suppose that a row vector $\mathbf{v} = (a(x), b(x)) \in R_{n,q}^2$, and arbitrary matrices $A \in R_{n,q}^{2\ell\times 2}$ of $2\ell \times 2$, define the external product of $\mathbf{v}$ and A as

$$\mathbf{v} \odot A = (\mathbf{v} \triangleleft G^{-1})A \in R_{n,q}^2;$$

it is a random vector; for $\mathbf{v} \triangleleft G^{-1}$ is a random vector of $1 \times 2\ell$. By definition, the external product satisfies the right distributive, namely, for arbitrary two matrices $A, B \in R_{n,q}^{(2\ell)\times 2}$ of $2\ell \times 2$, we have

$$\begin{aligned}\mathbf{v} \odot (A + B) &\equiv (\mathbf{v} \triangleleft G^{-1})(A + B)\\ &= (\mathbf{v} \triangleleft G^{-1})A + (\mathbf{v} \triangleleft G^{-1})B\\ &= \mathbf{v} \odot A + \mathbf{v} \odot B \pmod{(x^n + 1, q)}.\end{aligned}$$

2.4.3. GSW Ciphers

Let an n-bit secret key $s(x) = \sum_{i=0}^{n-1} s_i x^i$, where $s_i \in \{0, 1\}$, $RLWE$ sample $A \leftarrow R_{n,q}^{2\ell\times 2}$ (the rows of A are $RLWE$ samples) and a GSW cipher for $m(x) \in R_n$ is

$$GSW_s(m(x)) = A + m(x)G \in R_{n,q}^{2\ell\times 2};$$

according to the definition of $RLWE$ sample

$$A(-s(x), -1)^t \equiv \mathbf{w}(x) \pmod{(x^n + 1, q)},$$

where $\mathbf{w}(x) \in R_n^{2\ell}$, and $||\mathbf{w}(x)|| \leq \tau$; τ is the error size of GSW ciphers.

Lemma 3 ([7]). *Let $m_0(x), m_1(x) \in R_n$ be any two polynomials. For any $RE_s(m_0(x))$ with error size τ and any $GSW_s(m_1(x))$ with error size τ_1, we have*

$$RE_s(m_0(x)) \odot GSW_s(m_1(x)) = RE_s(m_0(x)m_1(x))$$

and $RE_s(m_0(x)m_1(x))$ which has an error size of at most $\tau||m_1(x)||_\infty + 4Bn\ell\tau_1$.

2.5. Bloom Filter

A Bloom filter [34] is a compact data structure for probabilistic set membership testing, and can insert and query data efficiently. The Bloom filter provides a time and space-efficient method to check whether there is an element in the set. A Bloom filter consists of a binary vector and a set of hash functions; b_j represents the j-th bit of the Bloom filter b and all elements of the empty Bloom filter are 0. Any Bloom filter b includes the three steps as follows:
$Create(\alpha)$: Create an empty Bloom filter with α bits; the hash function $\{h_i | 0 \leq i < \beta\}$ is:

$$h_i : \{0, 1\}^* \to \{0, \ldots, \alpha - 1\}.$$

$Add(x)$: Compute β hash values $g_i = h_i(x)$ of the element x using the hash function $h_i (0 \leq i < \beta)$. Set the Bloom filter cell with subscript g_i to 1.

$$g_i = h_i(x) \Longrightarrow b_{g_i} = 1$$

$Test(x)$: Test whether the element x is in the Bloom filter b. Compute β hash values $g_i = h_i(x)$ of the element x; if the β cells with subscript g_i are 1 ($b_{g_i} = 1$), then return 1 (true).

$$Test(x) = \bigwedge_{i=0}^{\beta-1} b_{h_i(x)} = \bigwedge_{i=0}^{\beta-1} b_{g_i} \tag{2}$$

The Bloom filter has a negligible false positive probability; $Test(x)$ will return 1, although x cannot be added to the Bloom filter. Given ω elements to be added and the expected maximum false positive probability 2^{-k}, the Bloom filter size α needs to satisfy:

$$\alpha \geq \frac{\omega k}{ln^2 2}.$$

A Bloom filter is widely used in cryptography. Bellovin and Cheswick [35] and Goh [36] implemented a securely document search using a Bloom filter. Raykov and Bellovin [37] realized a secure database query. Qiu L and Li Y [38] realized privacy data mining and BIP-0037 put forward the application of a Bloom filter in Bitcoin. Reference [39–41] realized the set intersection computing based on Bloom filters.

3. Homomorphic Operations

In SGFHE scheme, let any two LWE ciphers be $E_s(x_1)$ and $E_s(x_2)$ with $x_1, x_2 \in \{0,1\}$; one bootstrapping can compute three bit operations $E_s(x_1 \wedge x_2), E_s(x_1 \vee x_2)$ and $E_s(x_1 \oplus x_2)$; the scheme follows the approach in Ducas et al. [42] and Chillotti [43], but does not need to perform a key switch.

3.1. Key Generations

Let n be a power of 2, $n \geq 64$. Suppose that r can be divided by 8; $m = r/2$, $B = 35r^2n$,

$$r \geq 16n, q \geq nr, 1220r^4n^2 \leq Q < 1225r^4n^2 = B^2. \tag{3}$$

$$D_r = \lfloor r/4 \rfloor, D_q = \lfloor q/4 \rfloor, \widetilde{D}_Q = \lfloor Q/8 \rfloor, and\ G = \begin{pmatrix} 1 & 0 \\ B & 0 \\ 0 & 1 \\ 0 & B \end{pmatrix}.$$

Secret key Pick $\mathbf{s} \leftarrow \{0,1\}^n$ uniformly and randomly; let $s(x) = \sum_{i=0}^{n-1} s_i x^i$.
Public key $pk = (k_0(x), k_1(x)), k_0(x) \leftarrow R_{n,q}$,

$$k_1(x) \equiv k_0(x)s(x) + e(x) \pmod{x^n + 1, q},$$

where $e(x) \leftarrow R_n, ||e(x)||_\infty < D_q/(41n)$.
Bootstrapping key A bootstrapping key $bk = (C_0, C_1, \ldots, C_{n-1})$ can be generated as follows:
For $1 \leq i \leq n-1$ do:

(1) Pick $a_{ji} \leftarrow R_{m,Q}$, $1 \leq j \leq 4$;

(2) Pick $e_{ji}(x) \in R_m$, $||e_{ji}||_\infty \leq n$, $1 \leq j \leq 4$;

(3) Compute $b_{ji}(x) := a_{ji}(x)s(x) + e_{ji}(x) \pmod{x^m + 1, Q}$, $1 \leq j \leq 4$;

(4) Set $C_i := \begin{pmatrix} a_{1i}(x) & b_{1i}(x) \\ a_{2i}(x) & b_{3i}(x) \\ a_{3i}(x) & b_{2i}(x) \\ a_{4i}(x) & b_{4i}(x) \end{pmatrix} + s_i G \pmod{Q}$.

3.2. Bootstrapping Algorithm

Lemma 4. *Suppose that a bootstrapping key bk has an error size at most τ_1; r is divisible by 8 and $r \geq 16n, Q \geq \frac{n}{n-3}16Br^2\ell\tau_1$. Then, for any two LWE ciphers $E_s(x_i) = \boldsymbol{v}_i \in \mathbb{Z}_r^n \times \mathbb{Z}$, with error size $\leq D_r/4$ where $x_i \in \{0,1\}$ for $i = 1,2$, the bootstrapping algorithm in Algorithm 1 outputs random LWE ciphers $E_s(x_1 \wedge x_2), E_s(x_1 \vee x_2), E_s(x_1 \oplus x_2) \in \mathbb{Z}_r^n \times \mathbb{Z}_r$ all with error size $< n \leq D_r/4$ [7].*

Algorithm 1 Bootstrapping Algorithm: $BT_{bk}(\mathbf{v}_1, \mathbf{v}_2) \to \mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3$.

Input: $bk = (C_0, C_1, \ldots, C_{n-1}) \in R_{m,Q}^{2\ell\times 2}$: bootstrapping key;
$(\mathbf{v}_1, \mathbf{v}_2) \in \mathbb{Z}_r^n \times \mathbb{Z}_r$:$\mathbf{v}_i = E_s(x_i), x_1, x_2 \in \{0,1\}$;
Output: $E_s(x_1 \wedge x_2), E_s(x_1 \vee x_2), E_s(x_1 \oplus x_2) \in \mathbb{Z}_r^n \times \mathbb{Z}_r$;
1: Compute $\mathbf{u} := \mathbf{v}_1 + \mathbf{v}_2 = (u_0, u_1, \ldots, u_{n-1}, u_n) \in \mathbb{Z}_r^n \times \mathbb{Z}_r$;
2: $T := \{j \in \mathbf{Z} : -D_r \leq j \leq D_r\}, t(x) := \sum_{j\in T} x^j$;
3: $A := (0, t(x)x^{-u_n}\widetilde{D}_Q) \in R_{m,Q}^2$;
4: **for** k from 0 to n − 1 **do**
5: $A := A \odot (G + (x^{u_k} - 1)C_k)$;
6: **end for**
7: Let $A = (\sum_{i=0}^{m-1} a_i x^i, \sum_{i=0}^{m-1} b_i x^i)$. Set
$\mathbf{a}_1 := (Extract(a(x), 3m/4), \widetilde{D}_Q + b_{3m/4}) \in \mathbb{Z}_Q^n \times \mathbb{Z}_Q$;
$\mathbf{a}_2 := (Extract(a(x), m/4), \widetilde{D}_Q - b_{m/4}) \in \mathbb{Z}_Q^n \times \mathbb{Z}_Q$;
$\mathbf{a}_3 := \mathbf{a}_2 - \mathbf{a}_1 \in \mathbb{Z}_Q^n \times \mathbb{Z}_Q$;
8: **for** i from 1 to 3 **do**
9: $\mathbf{c}_i := \lfloor r\mathbf{a}/Q \rceil \in \mathbb{Z}_r^n \times \mathbb{Z}_r$;
10: **end for**
11: Return $\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3$;

3.3. Encryption Scheme

Lemma 5. *Suppose that $r = 2^{t+1}$; $(a(x), b(x)) \in R_{n,r}^2$ can be computed from Algorithm 2. Then for some $\omega_3(x), ||\omega_3(x)||_\infty \leq D_r/4$, so that*

$$2^{t-4}b(x) - s(x)a(x) \equiv \omega_3(x) + m(x)D_r \pmod{x^n + 1, r}.$$

Specifically, if $r = 16n$, then $(u, \boldsymbol{v})$ returned in Algorithm 2 has 6n bits and represents an RLWE cipher $RE_s(m(x))$, and the error size $< n$ [7].

Algorithm 2 Encryption with private key:$RE_s(m(x)) \to (u, \mathbf{v})$.

Input: n-bit secret key $s(x) = \sum_{i=0}^{n-1} s_i x^i$, $s_i \in \{0,1\}$;
n-bit message$m(x) = \sum_{i=0}^{n-1} m_i x^i$, $m_i \in \{0,1\}$;
$t := \lceil log_2(r) \rceil$, hence $2^t \leq r \leq 2^{t-1}$;
$P : \{0,1\} \to \{0,1\}^{n(t+1)}$;
Output: $(u, \mathbf{v}) \in \{0,1\}^n \times \{\{0,1\}^5\}^n$
1: $u \leftarrow \{0,1\}^n$, $a(x) := P(u, x) \in R_{n,r}$;
2: $\omega(x) \leftarrow R_n, ||\omega(x)||_\infty \leq D_r/8$, $b_1(x) := a(x)s(x) + \omega(x) + m(x)D_r \pmod{x^n + 1, r}$;
3: $b(x) = \sum_{i=0}^{n-1} b_i x^i := \lfloor b_1(x)/2^{t-4} \rfloor$;
4: $\mathbf{v} = (b_1, b_2, \ldots, b_{n-1}) \in (\{0,1\}^5)^n$;
5: return $(u, \mathbf{v})$;

Lemma 6. *Suppose that $r = 2^{t+1}$, $r \geq 16n$, $q \geq 4r$ and $n \geq 164$. Suppose that $RE_{pk}(m(x)) := (a(x), b(x)) \in R_{n,r}^2$ be any ciphertext output by Algorithm 3. Then $2^{t-5}b(x) - s(x)a(x) \equiv \omega_3(x) + m(x)D_r \pmod{x^n + 1, r}$ for some $\omega_3(x) \in R_n$ with $||\omega_3(x)||_\infty \leq D_r/4$.*

Specifically, if $r = 16n$*, then any ciphertext* $(a(x), b(x))$ *has* $n(10 + log_2(n))$ *bits and the error, that is, each coefficient of* $\omega_3(x)$*, is in* $(-n, n)$ *randomly [7].*

We can divide the data $\mathbf{x}$ into d blocks of length n. Let $N = dn$, $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_d) \in \{0,1\}^N$, $\mathbf{x}_k = (x_{k,0}, x_{k,1}, \ldots, x_{k,n-1}), \mathbf{x}_k \in \{0,1\}^n$. Each $\mathbf{x}_k$ can be expressed as a polynomial $\sum_{i=0}^{n-1} x_{k,i}x^i \in R_n$. Then—encrypted using the private-key scheme $\mathbf{c}_k = RE_s(\mathbf{x}_k), 1 \leq k \leq d$ by Algorithm 2—note that the cipher text size $\mathbf{c}_k$ is about 6N bits and then encrypted using the public-key scheme $\mathbf{c}'_k = RE_{pk}(\mathbf{x}_k), 1 \leq k \leq d$ by Algorithm 3; note that the cipher text size $\mathbf{c}_k{}'$ is about $N(10 + log_2(n))$. Homomorphic computing can be performed in three steps as follows:

Algorithm 3 Encryption under public key: $RE_{pk}(m(x)) \rightarrow (a(x), b(x)) \in R^2_{n,r}$.

Input: $pk = (k_0(x), k_1(x)), k_0(x) \leftarrow R_{n,q}$;
$m(x) = \sum_{i=0}^{n-1} m_i x^i$:$n$-bit message where each
$m_i \in \{0,1\}$;
$t := \lceil log_2(r) \rceil$;
Output: $(a(x), b(x)) \in R^2_{n,r} u(x) \leftarrow R_n$,
1: $u(x) \leftarrow R_n$, each coefficient random from $\{-1, 0, 1\}$;
2: $\omega_1(x) \leftarrow R_n, ||\omega_1(x)||_\infty \leq D_q/(41n)$;
3: $\omega_2(x) \leftarrow R_n, ||\omega_2(x)||_\infty \leq D_q/82$;
4: $a_1(x) := k_0(x)u(x) + \omega_1(x) \pmod{x^n + 1, q}$;
5: $b_1(x) := k_1(x)u(x) + \omega_2(x) + m(x)D_q \pmod{x^n + 1, q}$;
6: $a(x) := \left\lfloor \frac{r}{q} a_1(x) \right\rceil, b(x) := \left\lfloor \frac{r}{2^{t-5}q} b_1(x) \right\rceil$;
7: Return $(a(x), b(x))$

(1) Unpacking the *RLWE* ciphertexts $RE(\mathbf{x}_k)$ to get *LWE* ciphers in $\mathbb{Z}_r^n \times \mathbb{Z}_r$ for the bits of $\mathbf{x}$.

$$RE(\mathbf{x}_k) \xrightarrow{unpack} E_s(c_{k,i})$$

(2) Homomorphic computing of $f(\mathbf{x}) = \mathbf{y} = \{y_0, y_1, \ldots, y_M\} \in \{0,1\}^M$ on *LWE* ciphers.

$$f(\mathbf{x}) \xrightarrow{BT_{bk}} \{E_s(y_0), E_s(y_1), \ldots, E_s(y_M)\}$$

(3) Packing the *LWE* ciphers $\{E_s(y_0), E_s(y_1), \ldots, E_s(y_M)\}$ of function f into *RLWE* ciphers in $R^2_{n,r}$.

$$\{E_s(y_0), E_s(y_1), \ldots, E_s(y_M)\} \xrightarrow{pack} RE_s(\mathbf{y})$$

4. Privacy-Preserving Set Intersection

We abstract the privacy set intersection computation model as follows. The client C owns a set $\{\mathbf{c_1}, \ldots, \mathbf{c_v}\}$ of size v, and the server S holds a set $\{\mathbf{s_1}, \ldots, \mathbf{s}_\omega\}$ of size ω. After the end of the protocol, the client C only obtains the intersection $\{\mathbf{c_1}, \ldots, \mathbf{c_v}\} \bigcap \{\mathbf{s_1}, \ldots, \mathbf{s}_\omega\}$; however, the server cannot get any information for the input and the set intersection of the client (including the size of the intersection).

4.1. The Basic Two-Party Computing Protocol

The summary of basic private two-party intersection protocol is shown in Figure 1. The specific steps are as follows:

1. The client C encrypts the set with private key and sends ciphertexts to the server S.
2. The server S implements homomorphic computing with bootstrapping key and sends the result to the client C.

3. The client C decrypts and computes the intersection of the two sets; the server S cannot acquire any information about the input and output.

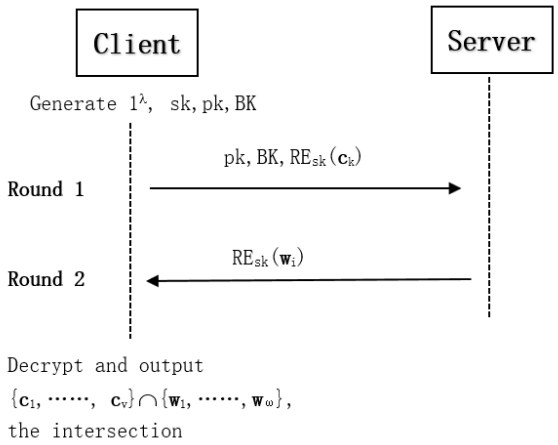

Figure 1. Summary of the intersection protocol.

Our basic two-party computing protocol is shown in Figure 2. At step $C \rightarrow S$, the client sends pk, bk and $RE_{sk}(\mathbf{c}_k)$ to the server. At step S, the server unpacks $RE_{sk}(\mathbf{c}_k)$ to get $E_{sk}(c_{k,j})$, unpacks RE_{pk} to get $E_{sk}(s_{i,j})$, samples $\mathbf{u} \in \{0,1\}^n$, calls bootstrapping operations to compute $E_{sk}(z_{k,i})$, computes LWE ciphers $E_{sk}(w_{i,j})$, packs the resulted LWE ciphers $E_{sk}(w_{i,j})$ into RLWE ciphers $RE_{sk}(\mathbf{w}_i)$ and sends them to the client. At step C, the client decrypts $RE_{sk}(\mathbf{w}_i)$ to get $\mathbf{w}_i$ and computes the intersection.

The basic two party computing protocol

Let $1 \le k \le v, 1 \le i \le \omega, 1 \le j \le n$;

$C \rightarrow S : pk, bk, RE_{sk}(\mathbf{c}_k)$;

$S : RE_{sk}(\mathbf{c}_k) \xrightarrow{unpack} E_{sk}(c_{k,j})$;

$RE_{pk}(\mathbf{s}_i) \xrightarrow{unpack} E_{sk}(s_{i,j})$;

$\mathbf{u} \xleftarrow{sample} \{0,1\}^n$;

$z_i = \bigwedge_{k=1}^{v} z_{k,i} = \bigwedge_{k=1}^{v} \bigvee_{j=1}^{n} (c_{k,j} \oplus s_{i,j}) \xrightarrow{BT_{bk}} E_{sk}(z_i)$;

$E_{sk}(w_{i,j}) = u_j E_{sk}(z_i) + s_{i,j}((0, D_r) - E_{sk}(z_i))$;

$\{E_{sk}(w_{i,1}), \ldots, E_{sk}(w_{i,n})\} \xrightarrow{pack} RE_{sk}(\mathbf{w}_i)$;

$S \rightarrow C : RE_{sk}(\mathbf{w}_i)$

$C : Dec(RE_{sk}(\mathbf{w}_i)) \Longrightarrow \mathbf{w}_i$;

$\{\mathbf{c}_1, \ldots, \mathbf{c}_v\} \bigcap \{\mathbf{w}_1, \ldots, \mathbf{w}_\omega\} \Longrightarrow \{\mathbf{c}_1, \ldots, \mathbf{c}_v\} \bigcap \{\mathbf{s}_1, \ldots, \mathbf{s}_\omega\}$.

Figure 2. The basic two-party computing protocol.

4.2. Correctness of the Basic Two-Party Computing Protocol

First, the correctness of SGFHE scheme has been proven.

Let $\mathbf{c}_k, \mathbf{s}_i$ be the set elements' binary representation of the client and server respectively. The insufficient bits are filled with 0s and we extend the length to n.

$$\mathbf{c}_k = \{c_{k,1}, \ldots, c_{k,n}\} = \{0,1\}^n, 1 \le k \le v$$

$$\mathbf{s}_i = \{s_{i,1}, \ldots, s_{i,n}\} = \{0,1\}^n, 1 \le i \le \omega$$

$$\mathbf{u} \xleftarrow{sample} \{0,1\}^n$$

$$z_{k,i} = \bigvee_{j=1}^{n} (c_{k,j} \oplus s_{i,j}) \tag{4}$$

If $z_{k,i} = 1$, then $\mathbf{c}_k \neq \mathbf{s}_i$; if $z_{k,i} = 0$, then $\mathbf{c}_k = \mathbf{s}_i$.

The server can acquire $E_{sk}(z_{k,i})$ by $RE_{sk}(\mathbf{c}_k) \xrightarrow{unpack} E_{sk}(c_{k,j}), RE_{pk}(\mathbf{s}_i) \xrightarrow{unpack} E_{sk}(s_{i,j})$ and call $(2n-1)$ bootstrapping operations, denoted by $z_{k,i} = \bigvee_{j=1}^{n} (c_{k,j} \oplus s_{i,j}) \xrightarrow{BT_{bk}} E_{sk}(z_{k,i})$.

Remark: RE represents RLWE cipher; E represents LWE cipher.

Let

$$z_i = \bigwedge_{k=1}^{v} z_{k,i}; \tag{5}$$

$E_{sk}(z_i) \in \mathbb{Z}_r^n \times \mathbb{Z}_r$ can be computed from $E_{sk}(z_{k,i})$ by implementing $(v-1)$ bootstrapping operations. Hence, implementing $(2n+v-2)$ bootstrapping operations by (6) can compute $E_{sk}(z_i)$.

$$z_i = \bigwedge_{k=1}^{v} z_{k,i} = \bigwedge_{k=1}^{v} \bigvee_{j=1}^{n} (c_{k,j} \oplus s_{i,j}) \xrightarrow{BT_{bk}} E_{sk}(z_i) \tag{6}$$

If $z_i = 1$, then $\mathbf{w}_i = \mathbf{u}$ is a random value with $\forall k, \mathbf{c}_k \neq \mathbf{s}_i$; if $z_i = 0$, then there $\exists k$ so that $\mathbf{c}_k = \mathbf{s}_i, \mathbf{w}_i = \mathbf{c}_k = \mathbf{s}_i$ is in the intersection. For $\mathbf{s}_i$ and $\mathbf{u}$, each bit

$$w_{i,j} = z_k \wedge u_j \oplus (1 - z_k) \wedge s_{i,j}$$

can be computed by

$$\mathbf{w}_i = \{w_{i,1}, \ldots, w_{k,n}\} = z_i \mathbf{u} \oplus (1 - z_i)\mathbf{s}_i \tag{7}$$

For plaintexts u_j and $s_{i,j}$, an LWE cipher of any bit $z_k \wedge u_j \oplus (1 - z_k) \wedge s_{i,j}$ can be computed as

$$u_j E_{sk}(z_i) + s_{i,j}(E_{sk}(1) - E_{sk}(z_k)),$$

which still has error size $< D_r/4$.

The *LWE* cipher is

$$E_{sk}(w_{i,j}) = u_j E_{sk}(z_i) + s_{i,j}(E_{sk}(1) - E_{sk}(z_k)). \tag{8}$$

The server can pack the resulted LWE ciphers $E_{sk}(w_{i,j})$ into RLWE ciphers $RE_{sk}(\mathbf{w}_i)$ and send them to the client.

In the end, the client decrypts $Dec(RE_{sk}(\mathbf{w}_i)) \Longrightarrow \mathbf{w}_i$ and computes the intersection $\{\mathbf{c}_1, \ldots, \mathbf{c}_v\} \bigcap \{\mathbf{w}_1, \ldots, \mathbf{w}_\omega\} \Longrightarrow \{\mathbf{c}_1, \ldots, \mathbf{c}_v\} \bigcap \{\mathbf{s}_1, \ldots, \mathbf{s}_\omega\}$.

4.3. Security Analysis of the Basic Two-Party Computing Protocol

We analyze the security of the protocol by comparing the real model and the ideal model. The real model is the actual implementation of the basic private intersection protocol and it is a trusted server for computing the intersection. The trusted server receives the input $\{\mathbf{c}_1, \ldots, \mathbf{c}_v\}$ of the client and the input $\{\mathbf{s}_1, \ldots, \mathbf{s}_\omega\}$ of the server, and will return the intersection with the client; however, the server cannot get any information about the output. The ideal model maintains all security evidence. In the semi-honest model, the participant's view includes its own input and the information received from other participants during the progression of the protocol. The simulator can use the participant's input and output to build a simulation that is computationally indistinguishable from the views. That proves that the participants cannot obtain any other information besides the inputs and outputs.

Theorem 1. *If SGFHE is held, then the basic two-party computing protocol can realize the private set intersection computing under the semi-honest model.*

Proof. In the protocol, the server cannot obtain any other information besides receiving the *RLWE* ciphers. Its view can only be simulated with ciphertexts and its security is based on IND-CPA security of *RLWE* scheme.

The client only receives the *RLWE* ciphers of the intersections and the random *RLWE* ciphers. Therefore, it just includes the output information of the set intersection and the view of simulator is only the output information of the set intersection. □

4.4. The Improvement of the Basic Two-Party Computing Protocol

In the basic two-party computing protocol, the server will return the ciphertexts of the intersection elements or the random ciphertexts, and computes the intersection by decrypting the ciphertexts. In our improvement protocol shown in Figure 3, we just need to determine whether $\mathbf{c}_k$ is in $\{\mathbf{s}_1, \ldots, \mathbf{s}_\omega\}$ without computing the ciphertexts of the intersection elements by the server. On the one hand, it can reduce the computational complexity; on the other hand, it will not reveal the size of the server set.

Improvement protocol

Let $1 \le k \le v, 1 \le i \le \omega, 1 \le j \le n$;
$C \to S : pk, bk, RE_{sk}(\mathbf{c}_k)$;
$S : RE_{sk}(\mathbf{c}_k) \xrightarrow{unpack} E_{sk}(c_{k,j})$;
$RE_{pk}(\mathbf{s}_i) \xrightarrow{unpack} E_{sk}(s_{i,j})$;
$z_{k,i} = \bigvee_{j=1}^{n} (c_{k,j} \oplus s_{i,j}) \xrightarrow{BT_{bk}} E_{sk}(z_{k,i})$;
$\{E_{sk}(z_{k,1}), E_{sk}(z_{k,2}), \ldots, E_{sk}(z_{k,\omega})\} \xrightarrow{pack} RE_{sk}(\mathbf{z}_k)$;
$S \to C : RE_{sk}(\mathbf{z}_k)$
$C : Dec(RE_{sk}(\mathbf{z}_k)) \Longrightarrow \mathbf{z}_k$;
$\mathbf{c}_k \notin \{\mathbf{c}_1, \ldots, \mathbf{c}_v\} \bigcap \{\mathbf{s}_1, \ldots, \mathbf{s}_\omega\}$ if $\mathbf{z}_k$ is all of 1, else $\mathbf{c}_k \in \{\mathbf{c}_1, \ldots, \mathbf{c}_v\} \bigcap \{\mathbf{s}_1, \ldots, \mathbf{s}_\omega\}$;

Figure 3. Improvement.

Let $\mathbf{c}_k, \mathbf{s}_i$ be the set elements' binary representations of the client and the server respectively. The insufficient bits are filled with 0s and we extend the length to n.

$$\begin{aligned} \mathbf{c}_k &= \{c_{k,1}, \ldots, c_{k,n}\} = \{0,1\}^n, 1 \le k \le v \\ \mathbf{s}_i &= \{s_{i,1}, \ldots, s_{i,n}\} = \{0,1\}^n, 1 \le i \le \omega \\ z_{k,i} &= \bigvee_{j=1}^{n} (c_{k,j} \oplus s_{i,j}) \end{aligned} \tag{9}$$

If $z_{k,i} = 1$, then $\mathbf{c}_k \neq \mathbf{s}_i$; if $z_{k,i} = 0$, then $\mathbf{c}_k = \mathbf{s}_i$.

The server can acquire $E_{sk}(z_{k,i})$ by $RE_{sk}(\mathbf{c}_k) \xrightarrow{unpack} E_{sk}(c_{k,j})$, $RE_{pk}(\mathbf{s}_i) \xrightarrow{unpack} E_{sk}(s_{i,j})$ and call $(2n-1)$ bootstrapping operations, denoted by $z_{k,i} = \bigvee_{j=1}^{n} (c_{k,j} \oplus s_{i,j}) \xrightarrow{BT_{bk}} E_{sk}(z_{k,i})$. The server packs *LWE* ciphers $E_{sk}(z_{k,i})$ to *RLWE* ciphers $RE_{sk}(\mathbf{z}_k)$ and sends them to the client.

$$\{E_{sk}(z_{k,1}), E_{sk}(z_{k,2}), \ldots, E_{sk}(z_{k,\omega})\} \xrightarrow{pack} RE_{sk}(\mathbf{z}_k),$$

the client decrypts $RE_{sk}(\mathbf{z}_k)$, if all $\mathbf{z}_k$ is 1, then

$$\mathbf{c}_k \notin \{\mathbf{c}_1, \ldots, \mathbf{c}_v\} \bigcap \{\mathbf{s}_1, \ldots, \mathbf{s}_\omega\};$$

else $\mathbf{c}_k \in \{\mathbf{c}_1, \ldots, \mathbf{c}_v\} \bigcap \{\mathbf{s}_1, \ldots, \mathbf{s}_\omega\}$.

In the protocol, the server cannot obtain any other information besides $RLWE$ ciphers and the view can only be simulated by the ciphertexts. Its security is based on IND-CPA security of $RLWE$ scheme.

The client acquires $z_{k,i}$ by (9), however, the probability of obtaining $s_{i,j}$ from $z_{k,i}$ and $\mathbf{c}_{k,j}$ is 2^{-n}, and it is negligible. The client only receives the output of the intersection; therefore, the view of simulator is just the output of the set intersection.

4.5. Two-Party Computing Protocol Based on a Bloom Filter

In this section, we construct a two-party protocol based on Bloom filter shown in Figure 4, in which the client C encrypts each bit of the Bloom filter with private key and sends it to the server S. The server S homomorphic computes $Test(\mathbf{s}_j)$ with the bootstrapping key of client C and sends it to the client. C will obtain the intersection of the two sets by decrypting, but the server cannot get any information about the input and output (including the size of the intersection).

Two-party computing protocol based on a Bloom filter

Let $1 \le k \le v, 1 \le i \le \omega, 1 \le j \le \omega, 1 \le t \le n$

$C : b = Create(\alpha)$;

$b.Add(\mathbf{c}_k)$;

$C \to S : pk, bk, \{h_i | 0 \le i < \beta\}, RE_{sk}(b)$;

$S : RE_{sk}(b) \xrightarrow{unpack} E_{sk}(b_1), \ldots, E_{sk}(b_\alpha)$;

$z_j = Test(s_j) = \bigwedge_{i=0}^{\beta-1} b_{h_i(\mathbf{s}_j)} \xrightarrow{BT_{bk}} E_{sk}(z_j)$; //$z_j$ =1 iff $\exists k, \mathbf{c}_k = \mathbf{s}_j$;

$\mathbf{u} \xleftarrow{sample} \{0,1\}^n$;

$E_{sk}(w_{j,t}) = s_{j,t} E_{sk}(z_j) + u_t((0, D_r) - E_{sk}(z_j))$;

$\{E_{sk}(w_{j,1}), \ldots, E_{sk}(w_{j,n})\} \xrightarrow{pack} RE_{sk}(\mathbf{w}_j)$;

$S \to C : RE_{sk}(\mathbf{w}_j)$;

$C : Dec(RE_{sk}(\mathbf{w}_j)) \Longrightarrow \mathbf{w}_j$;

$\{\mathbf{c}_1, \ldots, \mathbf{c}_v\} \bigcap \{\mathbf{w}_1, \ldots, \mathbf{w}_\omega\} \Longrightarrow \{\mathbf{c}_1, \ldots, \mathbf{c}_v\} \bigcap \{\mathbf{s}_1, \ldots, \mathbf{s}_\omega\}$.

Figure 4. Protocol based on a Bloom filter.

Let $\mathbf{c}_k, \mathbf{s}_i$ be the set elements' binary representations of the client and the server respectively. The insufficient bits are filled with 0s and we extend the length to n.

$$\mathbf{c}_k = \{c_{k,1}, \ldots, c_{k,n}\} = \{0,1\}^n, 1 \le k \le v.$$

$$\mathbf{s}_j = \{s_{j,1}, \ldots, s_{j,n}\} = \{0,1\}^n, 1 \le j \le \omega.$$

The client C constructs a Bloom filter $b = create(\alpha)$ and sends $pk, bk, RE_{sk}(b)$ to the server S.

$$z_j = Test(s_j) = \bigwedge_{i=0}^{\beta-1} b_{h_i(s_j)} \tag{10}$$

According to (10), input $E_{sk}(b_1), \ldots, E_{sk}(b_\alpha)$,

$$E_{sk}(z_j) = \bigwedge_{i=0}^{\beta-1} E_{sk}(b_{h_i(s_j)}).$$

Call $(\beta - 1)$ bootstrapping operations to obtain $E_{sk}(z_j)$, denoted by

$$z_j = Test(s_j) = \bigwedge_{i=0}^{\beta-1} b_{h_i(\mathbf{s}_j)} \xrightarrow{BT_{bk}} E_{sk}(z_j).$$

$$\mathbf{w}_j = \{w_{j,1}, \ldots, w_{j,n}\} = z_j \mathbf{s}_j \oplus (1 - z_j)\mathbf{u} \tag{11}$$

If $z_j = 1$, then there $\exists k$ such that $\mathbf{c}_k = \mathbf{s}_j$, and computing $\mathbf{w}_j = \mathbf{s}_j$ by (11); similarly, if $z_j = 0$, then $\forall k$ such that $\mathbf{c}_k \neq \mathbf{s}_j$, and computing $\mathbf{w}_j = \mathbf{u}$ by (11). For plaintexts $\mathbf{s}_j$ and $\mathbf{u}$, each bit can be computed by (11),

$$w_{j,t} = z_j \wedge s_{j,t} \oplus (1 - z_j) \wedge u_t, 1 \leq t \leq n.$$

The corresponding *LWE* cipher is

$$E_{sk}(w_{j,t}) = s_{j,t} E_{sk}(z_j) + u_t(E_{sk}(1) - E_{sk}(z_j)). \tag{12}$$

The correctness and security of the two-party computing protocol based on Bloom filter is similar to the basic two-party computing protocol. Please refer to Sections 4.2 and 4.3.

5. Conclusions

We constructed the set intersection two-party computing protocols based on a fully homomorphic encryption scheme. The protocols are simple and only need two rounds of communication, and the security is based on *RLWE* and *LWE* problems in the semi-honest model. The ciphertext extension of the protocols is small so that the protocols have strong practicability. Furthermore, we can extended the set intersection protocol by outsourcing computing under the malicious model. The limitation of our schemes is they are two-party protocols. In future work, we shall extend them to multi-party protocols. The disadvantage of the private set intersection protocols is they are not efficient enough due to bottleneck the bootstrapping operation. On the theoretical side, with the development of fully homomorphic encryption technology, its performance has been greatly improved, but the efficiency of it is still worthy of in-depth study. The bottleneck of the SGFHE scheme is its bootstrapping operation; therefore, its parallelization and hardware implementation will be further studied to improve the overall efficiency of the protocol.

Author Contributions: Conceptualization, Y.C., C.T. and Q.X.; methodology, Y.C. and C.T.; validation Y.C., C.T. and Q.X.; writing-original draft preparation, Y.C., Q.X. and C.T.; writing-review and editing, Y.C. and Q.X. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the Foundation of National Natural Science of China under grant 61772147, in part by Guangdong Province Natural Science Foundation of major basic research and Cultivation project under grant 2015A030308016, in part by Project of Ordinary University Innovation Team Construction of Guangdong Province under grant 2015KCXTD014, in part by Basic Research Major Projects of Department of education of Guangdong Province under grant 2014KZDXM044, in part by Collaborative Innovation Major Projects of Bureau of Education of Guangzhou City under grant 1201610005 and in part by the Key-Area Research and Development Plan of Guangdong province under grant 2019B020215004.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Rvest, R.L.; Adieman, L.; DERtouzos, M.L. On data banks and privacy homomorphisms. *Found. Secur. Comput.* **1978**, *4*, 169–180.
2. Craig, G. Fully Homomorphic Encryption Using Ideal Lattices. In Proceedings of the Annual ACM Symposium on Theory of Computing, Bethesda, MD, USA, 31 May–2 June 2009; pp. 169–178. [CrossRef]
3. Van Dijk, M.; Gentry, C.; Halevi, S.; Vaikuntanathan, V. Fully homomorphic encryp-tion over the integers. *IACR Cryptol. Eprint Arch.* **2009**, 616. [CrossRef]
4. Brakerski, Z.; Vaikuntanathan, V. Efficient fully homomorphic encryption from (standard) LWE. *SIAM J. Comput.* **2014**, *43*, 831–871. [CrossRef]
5. Brakerski, Z.; Vaikuntanathan, V. Fully homomorphic encryption from ring-LWE and security for key dependent messages. In Proceedings of the Advances in Cryptology-CRYPTO 2011, Santa Barbara, CA, USA, 14–18 August 2011; Springer: Berlin/Heidelberg, Germany, 2011; pp. 505–524. [CrossRef]
6. Gentry, C.; Sahai, A.; Waters, B. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *Advances in Cryptology-CRYPTO 2013*; Springer: Berlin/Heidelberg, Germany, 2013; pp. 75–92. [CrossRef]

7. Gao, S. Efficient Fully Homomorphic Encryption Scheme. Cryptology ePrint Archive: Report 2018/637. 2018. Available online: https://eprint.iacr.org/2018/637 (accessed on 28 October 2020).
8. Chillotti, I.; Gama, N.; Georgieva, M. Improving TFHE: Faster Packed Homomorphic Operations and Effcient Circuit Bootstrapping. Cryptology ePrint Archive: Report 2017/ 430. 2017. Available online: https://eprint.iacr.org/2017/430 (accessed on 28 October 2020).
9. Oded, G.; Micali, S.; Avi, W. How to play ANY mental game. In Proceedings of the 19th Annual ACM Symposium on Theory of Computing, New York, NY, USA, 25–27 May 1987; pp. 218–229. [CrossRef]
10. Ion, M.; Kreuter, B.; Nergiz, E. Private Intersection-Sum Protocol with Applications to Attributing Aggregate Ad Conversions. Cryptology ePrint Archive: Report 2017/738. 2017. Available online: https://eprint.iacr.org/2017/738 (accessed on 28 October 2020).
11. Ion, M.; Kreuter, B.; Erhan, A. On Deploying Secure Computing Commercially: Private Intersection-Sum Protocols and their Business Applications. Cryptology ePrint Archive: Report 2019/723. 2019. Available online: https://eprint.iacr.org/2019/723. (accessed on 28 October 2020).
12. Prasetyo, H.; Guo, J.M. A Note on Multiple Secret Sharing Using Chinese Remainder Theorem and Exclusive-OR. *IEEE Access* **2019**, *7*, 37473–37497. [CrossRef]
13. Yang, Q.; Peng, G.; Gasti, P. MEG: Memory and Energy Efficient Garbled Circuit Evaluation on Smartphones. *IEEE Trans. Inf. Forensics Secur.* **2019**, *14*, 913–922. [CrossRef]
14. Zhang, Z.; Zhang, F.G. Garbled Circuits and Indistinguishability Obfuscation. *J. Cryptologic Res.* **2019**, *6*, 541–560. [CrossRef]
15. Qin, H.; Wang, H.; Wei, X. Privacy-Preserving Wildcards Pattern Matching Protocol for IoT Applications. *IEEE Access* **2019**, 1. [CrossRef]
16. Gama, M.; Mateus, P.; Souto, A. A Private Quantum Bit String Commitment. *Entropy* **2020**, *22*, 272. [CrossRef]
17. Zhao, C.; Zhao, S.N.; Jia, Z.T. Advances in Practical Secure Two-party Computation and Its Application in Genomic Sequence Comparison. *J. Cryptol. Res.* **2019**, *6*, 194–204.
18. Hazay, C.; Lindell, Y. Efficient Protocols for Set Intersection and Pattern Matching with Security against Malicious and Covert Adversaries. *J. Cryptol.* **2010**, *23*, 422–456. [CrossRef]
19. Cristofaro, E.D.; Lu, E.; Tsudik, Y.A. Gene Efficient Techniques for Privacy-Preserving Sharing of Sensitive Information. Cryptology ePrint Archive: Report 2011/113. 2011. Available online: https://eprint.iacr.org/2011/113 (accessed on 28 October 2020).
20. Saracevic, M.; Adamovic, S.; Miskovic, V.; Macek, N.; Sarac, M. A novel approach to steganography based on the properties of Catalan numbers and Dyck words. *Future Gener. Comput. Syst.* **2019**, *100*, 186–197. [CrossRef]
21. Saracevic, M.; Adamovic, S.; Bisevac, E. Applications of Catalan numbers and Lattice Path combinatorial problem in cryptography. *Acta Polytech. Hung.* **2018**, 15, 91–110.
22. Coppolino, L.; D'Antonio, S.; Formicola, V.; Mazzeo, G.; Romano, L. VISE: Combining Intel SGX and Homomorphic Encryption for Cloud Industrial Control Systems. *IEEE Trans. Comput.* **2020**, *99*. [CrossRef]
23. Lindell, Y.; Pinkas, B. Secure Multiparty Computation for Privacy-Preserving Data Mining. *J. Priv. Confidentiality* **2009**. [CrossRef]
24. Michael, L.; Nejdl, W.; Papapetrou, O.; Siberski, W. Improving Distributed Join Efficiency with Extended Bloom Filter Operations. In Proceedings of the 21st International Conference on Advanced Networking and Applications, Niagara Falls, ON, Canada, 21–23 May 2007.
25. Naor, M.; Pinkas, B. Oblivious Transfer and Polynomial Evaluation. In Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing, Atlanta, GA, USA, 1–4 May 1999.
26. Freedman, M.J.; Ishai, Y.; Pinkas, B. Keyword Search and Oblivious Pseudorandom Functions. In *Second International Conference on Theory of Cryptography*; Springer: Berlin/Heidelberg, Germany, 2005. [CrossRef]
27. Chaum, D.; Rivest, R.L.; Sherman, A.T. Blind Signatures for Untraceable Payments. *Adv. Cryptol.* **1983**. [CrossRef]
28. Florian, K. Outsourced private set intersection using homomorphic encryption. *ACM Symp. Inf.* **2012**, 85–86. [CrossRef]
29. Nojima, R.; Kadobayashi, Y. Cryptographically Secure Bloom-Filters. *Trans. Data Priv.* **2009**, *2*, 131–139.
30. Shen, L.; Chen, X.; Shi, J.; Hu, L. Survey on Private Preserving Set Intersection Technology. *J. Comput. Res. Dev.* **2017**, *54*, 2153–2169. [CrossRef]

31. Regev, O. On lattices, learning with errors, random linear codes, and cryptography. In Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, 22–24 May 2005. [CrossRef]
32. Regev, O. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM* **2009**, *34*. [CrossRef]
33. Lyubashevsky, V.; Peikert, C.; Regev, O. On ideal lattices and learning with errors over rings. In *Cryptology-EUROCRYPT 2010*; Springer: Berlin/Heidelberg, Germany, 2010; p. 6110.
34. Bloom, B.H. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM* **1970**, *13*, 422–426. [CrossRef]
35. Bellovin, S.; Cheswick, W. Privacy-Enhanced Searches Using Encrypted Bloom Filters. Cryptology ePrint Archive: Report 2004/022. 2004. Available online: https://eprint.iacr.org/2004/022 (accessed on 28 October 2020).
36. Goh, E. Secure Indexes. Cryptology ePrint Archive: Report 2003/216. 2003. Available online: https://eprint.iacr.org/2003/216 (accessed on 28 October 2020).
37. Raykova, M.; Vo, B.; Bellovin, S.; Malkin, T. Secure Anonymous Database Search. In Proceedings of the ACM Cloud Computing Security Workshop, Chicago, IL, USA, 13 November 2009; pp. 115–126. [CrossRef]
38. Qiu, L.; Li, Y.; Wu, X. Preserving privacy in association rule mining with bloom filters. *J. Intell. Inf. Syst.* **2007**, *29*, 253–278. [CrossRef]
39. Debnath, S.K.; Dutta, R. Secure and Efficient Private Set Intersection Cardinality Using Bloom Filter. *Int. Inf. Secur. Conf.* **2015**, *9290*, 209–226. [CrossRef]
40. Changyu, D.; Liqun, C.; Zikai, W. When private set intersection meets big data: An efficient and scalable protocol. In Proceedings of the ACM Conference on Computer and Communications Security, Berlin, Germany, 4–8 November 2013; ACM: New York, NY, USA, 2013; pp. 789–800. [CrossRef]
41. Egert, R.; Fischlin, M.; Gens, D. Privately Computing Set-Union and Set-Intersection Cardinality via Bloom Filters. *Eur. J. Oper. Res.* **2015**, *139*, 371–389. [CrossRef]
42. Ducas, L.; Micciancio, D. FHEW: Bootstrapping homomorphic encryption in less than a second. In Proceedings of the Advances in Cryptology-EUROCRYPT 2015, Sofia, Bulgaria, 26–30 April 2015; Springer: Berlin/Heidelberg, Germany, 2015; Part I, Volume 9056, pp. 617–640. [CrossRef]
43. Chillotti, I.; Gama, N.; Georgieva, M.; Izabachéne, M. Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. In Proceedings of the Advances in Cryptology-ASIACRYPT 2016: 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, 4–8 December 2016; Springer: Berlin/Heidelberg, Germany, 2016; Part I, Volume 10031, pp. 3–33. [CrossRef]