Private Set Intersection from TFHE for Cloud Computing Scenarios

Jiayu Qiang[1,2*], Yi Deng[1,2,3*]
[1]State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China
[2]School of Cyber Security, University of Chinese Academy of Sciences, Beijing 100093, China
[3]State Key Laboratory of Cryptology, P.O.Box 5159, Beijing, 100878, China

***Abstract*—In most scenarios of Private Set Intersection (PSI) computed on a cloud server, the client has a smaller set size and lower computation ability than that of the cloud server, which is known as the unbalanced setting. We use Torus Fully Homomorphic Encryption (TFHE) for the first time instead of the leveled ones to construct a PSI protocol. More precisely, we mainly focus on an adaptive and dynamic setting since the server may provide services to multiple clients at the same time and its data set is updated in real time.**

We use TFHE to construct an adaptive PSI for unbounded items with a lower communication complexity of $O(|Y|)$ than [19](CCS17), where Y is the length of client's sets) . TFHE support arbitrary depth of homomorphic operations, which avoids those optimizations[19] made to reduce the depth of the circuit, resulting in additional computation and communication complexity. We propose a basic protocol that can efficiently compute the intersection with small items and then we apply a partition technique to our full protocol in order to support unbounded items. We also achieve a flexible dynamic protocol by adjusting our parameters into an adaptive setting, which can further reduce the communication cost of our PSI protocol, especially in cloud computing scenarios mentioned above.

***Index Terms*—private set intersection, fully homomorphic encryption, cloud computing**

I. Introduction

In the client-server setting in PSI with a cloud server, client with low computation ability holds a small data set while the server is opposite. Thus, the best strategy is to reduce the workload of the client by giving all of the computation tasks to could server. Since the data size is huge regularly, most scenarios of cloud computing is not a immediate use case and computation complexity is seen as a considerable factor. Customers may want to maintain the privacy of their data while getting computing results efficiently so that cloud computing is often used as a powerful component of protocols dealing with privacy information.

As we discussed above, a PSI that can deal with dynamic data sets and with low communication cost is needed under this scenario. Further more, the cloud server may prefer an adaptive strategy when providing services to the clients whose set size are quietly different.

*Corresponding author (qiangjiayu@iie.ac.cn, deng@iie.ac.cn)

The study of constructing a PSI with a leveled FHE has already mentioned in [19]. We now give another feasible approach to construct a PSI based on a boolean circuit FHE with even less communication complexity than [19]. Our motivation of applying a boolean circuit FHE in our PSI protocol is explained as follow:

a FHE schemes operate on the ciphertexts directly so that only clients need to send their small encrypted data sets to the server instead of both of them calculating the intermediate results and interacting with each other. We take advantage of this feature in FHE and the study in [19] as a fundamental step to reduce our communication.

b We further reduce the communication complexity by replacing the underlying FHE scheme from a leveled one to a boolean circuit FHE. The leveled FHE applied in [19] has to employ some optimizations(which will increase communication) on circuit depth since the depth in those scheme is fixed by the pre-setting parameter. We will demonstrate that the FHE we used in our protocol can completely avoid this problem and achieve extra advantage in following sections. Moreover, many optimizations can be used in our PSI, such as Cuckoo hashing. However, it would still have a slightly effect on computation efficiency, but the running time is still acceptable while the scenario our protocol focused on is computing on a cloud server and without immediacy.

c We can achieve characteristics of adaptive and dynamic when applying the boolean circuit FHE while the leveled FHE based PSI is more suitable for a static data set. Thus our protocol perform well in the setting we mentioned above. Furthermore, computing on arbitrary depth of circuit and unbounded items is also can be achieved combining with some optimizations.

We give brief introductions to private set intersection(PSI) and fully homomorphic encryption(FHE) in following two subsections, respectively, and summarize our contributions in the final subsection.

A. Private Set Intersection

As the discussion above, private set intersection (PSI) allows two parties with respective input sets X and Y to compute the intersection of the two sets without revealing anything else and is applicable to many scenarios with operating privacy-sensitive information.

Over past few years, many cryptographic components and techniques were employed to PSI, such as garbled circuit[1], oblivious transfer (OT)[2–5], FHE[19, 20] and oblivious polynomial evaluation (OPE)[7, 8], so as to meet requirements of various practical scenarios. Many protocols begin to focus on specific settings due to various techniques supporting PSI to achieve higher efficiency. For example, [9, 20] are designed for communication setting especially with unequaled data set, such as mobile devices [9], while [6, 10] pay more attention to computation efficiency.

B. Fully Homomorphic Encryption

Fully Homomorphic Encryption (FHE) allows to directly evaluate on ciphertext without accessing to the decryption key instead of operating decrypted data. The first fully homomorphic encryption scheme based on ideal lattices was proposed by Gentry in 2009[13] while the notion of homomorphic encryption could date back to 1978 by Rivest [14].

Several years later, many researchers proposed a list of leveled FHE based on arithmetic circuit[15–18] which were widely used in processing private information, such as MPC, Neural Networks and even PSI[19, 20]. Leveled FHE should process bootstrapping procedure before the noise gets too big to destroy the message encoded in the ciphertexts, but the bootstrapping has a high computation cost[18]. Thus, many applications try to avoid bootstrapping to get high efficiency by retaining enough space and using other techniques to dealing with noise flooding problem[11, 19, 20].

Conversely, TFHE schemes has fast bootstrapping procedure after each gate, which the cost is down to 13ms in [23]. At a high level, more studies should be done to develop the applications of Boolean circuit FHE since it is a universal construction and and has a better way to manage noise.

C. Our Contributions

In this paper, we propose an efficient PSI protocol based on Boolean FHE scheme for the first time. More precisely, we use torus fully homomorphic encryption(TFHE)[22–24] and variation of its integer comparison algorithm[21] in our work. We reduce pairwise-comparisons and achieve a dynamic feature via Cuckoo hashing technique which is slightly different from [19]. We also use a partition strategy to make our protocol support to deal with unbounded items.

According to [19], there is a limitation producing from the pre-setting circuit depth which lead to large parameter setting and inability of computing arbitrary depth of circuit. Thus, several optimization need to be applied to ensure the circuit depth is in the legal range so that leading to additional communication even related to the size of the larger set. We completely avoid this extra cost by constructing our PSI based on a Boolean FHE instead of leveled ones and achieve the ability of computing arbitrary circuit.

We also achieve a flexible dynamic protocol by adjusting our parameters into an adaptive setting. The communication cost from the client would dynamically depending on its own set size which avoiding wastes from a large presetting. Besides, benefiting from the underlying Boolean circuit FHE, the hash table of the larger set(which refers to server’ s set according to above setting), can adding data dynamically without considering the problem of overload. While in [19], the protocol fall when the hash bin is overload, so that a large parameter setting is needed to reduce it to a small probability.

In summary, our contributions are as follows:

- Construct a basic protocol via the variation of integer comparison based on TFHE, and realize faster intersection of small items.
- Propose a strategy of *dynamic hashing to bins*, which support sender(cloud server) updating the database in real time and serving receivers(client) of different data set sizes at the same time.
- Construct a full protocol to support items of arbitrary length by using *hashing to shorter strings* and *partitioning* technique.
- Take advantage of the efficient bootstrapping procedure to achieve arbitrary depth of homomorphic operations which results in communication complexity of $O(|Y|)$.

II. Prelimiatities

In this section, we first start by introducing necessary knowledge about cyclotomic ring and notations used in this paper. Then, a description of TFHE is given especially some terms and the details of sub algorithm. Finally, we recall the idea of constructing PSI with FHE in CLR17[19] and explain the features of this type of PSI.

A. Cyclotomic ring and notations

For a power of 2 denoted by 2^N, let $\Phi_{2N}(x) = x^N + 1$ be the $2N$-th cyclotomic polynomial. We denote $\mathbb{Z}_N[x]$ as cyclotomic ring $\mathbb{Z}(x)/\Phi_{2N}$. For a set S and a positive integer n, we write S^n be a n dimensional vector which each element is belong to S.

Let $[a]$ denotes a set $\{1, 2, \cdots, a\}$ when a is a positive integer, and $i \leftarrow S$ means picking a random element from this set S. The security parameter is denoted as λ. The hash table from the set X is denoted by $\mathfrak{T}_X$.

B. TFHE scheme

As in [23], Chillotti et al. use Learning with Errors(LWE) and Ring Learning With Errors (RLWE) problem over torus $\mathbb{T} = \mathbb{R}/\mathbb{Z}$ (i.e., the real modulo 1) for the construction of fully homomorphic encryption scheme.

Chillotti et al. also refer to LWE over torus as LWE. LWE problem is hard to solve even for quantum computers. The LWE assumption states that it's hard to distinguish between $(\mathbf{a}, \mathbf{as} + e) \in \mathbb{T}^{n'+1}$ and the uniform distribution over $\mathbb{T}^{n'+1}$ without the knowledge of $\mathbf{s}$, for $\mathbf{a} \leftarrow \mathbb{T}^{n'}$, $\mathbf{s} \leftarrow \{0,1\}^{n'}$, $e \leftarrow \chi$, χ is a continuous gaussian distributions over $\mathbb{R}$.

As a variant of LWE, the RLWE problem is also extensively used for the reason of ciphertexts based on it has a larger message space. Let $\mathcal{R}_N = \mathbb{R}_N[x]/\mathbb{Z}_N[x]$ with $\mathbb{R}_N[x] = \mathbb{R}[x]/(x^N + 1)$ and $\mathbb{Z}_N[x] = \mathbb{Z}[x]/(x^N + 1)$, N is a power of 2. The RLWE assumption states that it's hard to distinguish between $(a, as + e) \in \mathcal{R}_N^2$ and the uniform distribution over $\mathcal{R}_N^2$ without the knowledge of s, for $a \leftarrow \mathcal{R}_N$, $s \leftarrow \mathbb{Z}_2[x]/x^N + 1$, $e \leftarrow \chi$, χ is a distribution over $\mathbb{R}_N[x]$.

The TFHE scheme has an efficient bootstrapping that enables to reduce errors after ciphertext calculation.

Like in [21], we define the encryption scheme as follow:

- **TFHE.Setup**(1^λ): Inputs a security parameter λ to obtain public *paras* including the length of secret key n, and returns $\mathbf{s} \leftarrow \{0,1\}^{n'}$.
- **TFHE.Encrypt**$(\mathbf{s}, \mathtt{m})$: Inputs message $\mathtt{m}$ and secret key $\mathbf{s}$. Sampling $\mathbf{a} \leftarrow \mathbb{T}^{n'}$ and $e \leftarrow \chi$ where χ is continuous gaussian distributions over $\mathbb{R}$, and return $\vec{c} = (\mathbf{a}, b)$ where $b = \mathbf{as} + e + \frac{m}{2B}$;
- **TFHE.Decrypt**$(\mathbf{s}, ct)$: Computes $\vec{c}(-\mathbf{s}, 1)$ mod 1 in $[-\frac{1}{2}, \frac{1}{2}[$ and return $\lfloor 2B \cdot x \rceil$.

Unless special instructions in the rest of the paper, LWE ciphertext is belong to $\mathbb{T}^{n'+1}$ while RLWE ciphertext is belong to $\mathcal{R}_N^2$. We denotes them as $\vec{c}$ and $\mathtt{ct}$, respectively.

We define two subprocedure called BindRotate and Extract as follow:

- **BlindRotate**: inputs an LWE ciphertext $\vec{ct}$ encrypted with $\mathbf{s}$ and a bootstrapping key bk, returns an RLWE ciphertext encryption of $x^{b-\overline{\mathbf{as}}}$ where $b = \lfloor 2N \cdot b \rceil$ and $\overline{\mathbf{a}} = \lfloor 2N \cdot \mathbf{a} \rceil$.
- **Extract**: Inputs an RLWE ciphertext of a polynomial $p(x)$ and return an LWE ciphertext of $p(0)$.

We use homomorphic AND and OR gates as follow:

- **HomAND**: input two LWE ciphertext $\vec{c}_1$ and $\vec{c}_2$, output Bootstrap$((0, -\frac{1}{8}) + \vec{c}_1 + \vec{c}_2)$.
- **HomOR**: input two LWE ciphertext $\vec{c}_1$ and $\vec{c}_2$, output Bootstrap$((0, \frac{5}{8}) - \vec{c}_1 - \vec{c}_2)$.

C. Overview of private set intersection protocol via homomorphic encryption

In this subsection, we recall the first protocol based on FHE proposed in [19] in order to achieve a lower communication cost. In practical scenarios, the party who wants to get the result often seems to be a client with small set size and low computation ability, while the other one can be seen as a powerful server who own a large set. FHE seems to be a directly solution in this setting since it only requiring the client sending its small encrypted data set and waiting for the result from server, instead of both sides interacting their intermediate results. Thus, protocol introduced in [19] reduce the communication cost significantly in such unbalanced setting.

For more detail, they invite a leveled FHE scheme mentioned in [16] and set enough parameters to cover the increasing noise after each homomorphic operation as we discussed in I-B. They also adopt several optimization, such as Batching and Cuckoo hashing to deduce the comparison-with-pairs, windowing and partitioning to reduce the circuit depth.

Since bootstrapping procedure of [16] is inefficient and leading to their abjuration, only limited depth of circuit can be compute in [19] because of the growing noise. Later in [20], they use OPRF to resolve this problem but leading to two more rounds of interaction. Also, due to the concrete property of a FHE based on arithmetic circuit, [19] and even [20] cannot dealing with a circuit deeper than they pre-defined.

III. Basic Private set intersection protocol and its optimization

In this section, we propose a basic PSI protocol based on a torus fully homomorphic encryption(TFHE). A strategy of *dynamic hashing to bins* which would be applied to improve the efficiency of the protocol is proposed.

A. Basic PSI protocol based on TFHE

In many practical scenarios, there are only two parts consists of sender and receiver, where sender's set is significantly larger than receiver's as [9, 19]. The unbalanced PSI protocols based on homomorphic encryption has the lowest communication cost but large computational overhead. In fact, the communication is most likely to be a limited factor in the client-server model. As a server, sender performs most of the operations in consideration of cloud server has a power device.

As in [19], X denotes the sender's set while Y denotes receiver's set, whose length are s_x and s_y with $s_x \gg s_y$, respectively. The length of each item in both sets is σ. We start with a basic scheme that the parameter σ is small. We only consider the feasibility instead of efficiency without any optimization this subsection.

We construct our basic PSI protocol based on the study of [21], which describes an integer comparison algorithm for unbounded inputs based on TFHE. We reconstruct

Algorithm 1 Basic private set intersection protocol based on TFHE homomorphic encryption

```
Input: Receiver inputs set $Y$ while sender inputs set $X$, and whose size is $s_y$ and $s_x$, respectively. $s_y$, $s_x$, $\sigma$ and $\lambda$ are public parameters.
output: Receiver outputs $X \cap Y$; sender outputs $\perp$.
  1) Setup: Receiver and sender run RLWE.Setup($1^\lambda$) to obtain public $paras$. We suppose $2^{(\sigma-1)} < n$. Receiver keeps its secret key $\mathbf{s}$ and $s$.
  2) Encryption: Receiver encrypts each negative element $-y_i$ from set $Y$ in the exponent of $X$, (e.g. run RLWE.Encrypt($s$, $x^{-9}$) when encrypted item is 9), and then sends $s_y$ RLWE ciphertexts ($\mathtt{ct}_1$, $\mathtt{ct}_2$, $\cdots$, $\mathtt{ct}_{s_y}$) to sender.
  3) Intersection: For each $\mathtt{ct}_j$, sender
     (a) let $p(x) = x^{x_1} + x^{x_2} + \cdots + x^{x_{s_x}}$ (if two identical $x_j$, $j \in [s_x]$ exist, remains one of them), and compute $\vec{p_i} = \text{Extract}(p(x) \cdot \mathtt{ct}_j)$.
     (b) return $s_y$ LWE ciphertexts ($\vec{p}_1, \vec{p}_2, \cdots, \vec{p}_{s_y}$) to receiver.
  4) Decryption: Receiver decrypts each LWE ciphertext and outputs $X \cap Y = \{y_i : \text{TFHE.Decrypt}(\mathbf{s}, \vec{p_i}) = 1\}$.
```

the $p(x)$ in Extract algorithm in order to fit our step of interaction.

A process of PSI could be considered as a great number of one-to-one comparisons. However, It is not clever to run $s_x s_y$ comparison algorithm on account of PSI would be inefficient in this way. We use the idea of [21] but a completely different strategy.

We start with an introduction of RLWE encryption in our protocol, the ciphertext parameters are the same as TFHE's RLWE, which is defined as follow:

- **RLWE.Encrypt**(s, $\mathtt{m}$): Inputs message $\mathtt{m}$ and secret key s. Randomly sample $a \leftarrow \mathcal{R}_N$ and $e \leftarrow \chi$ where χ is a distribution over $\mathbb{R}_N[X]$, and return $\mathtt{ct} = (a, b)$ where $b = as + e + \frac{\mathtt{m}}{2n}$;

The detailed basic PSI protocol is shown in Algorithm 1. We now using the following description to explain that Algorithm 1 is a valid protocol.

Correctness. The input of the Extract algorithm is a RLWE ciphertext and a polynomial $p(x)$. We define a $p(x)$ which contains sender items so that output of Extract algorithm is a ciphertext of 0 if receiver's item is the same as one of the sender's, which also could refer to [21].

Semi-honest Security of Basic Protocol. Our basic scheme can security compute the intersection set in the semi-honest security model. In this setting, a semi-honest receiver has no additional ability to conduct analysis on its outputs or to send a distorted message. In the view of receiver, the TFHE scheme we based on has a security of IND-CPA[28] so that the encrypted data under RLWE is pseudorandom towards the sender. According to the circuit privacy under semi-honest model mentioned in [11], the receiver learn nothing from the output ciphertext for each item which produced by sender' s circuit.

As the discussion in I-B, the FHE we used in our protocol has a efficient bootstrapping algorithm(where a leveled FHE hasn't) after single gate operation, which allows to sanitize the ciphertext based on successive iterations of bootstrapping. Sanitation for ciphertexts is a technique first proposed by L. Ducas. et.al[11] and is designed to replace the flooding strategy and allows to take much smaller parameters which contributes to keep our protocol flexible.

B. Dynamic Hashing to bins

Hashing to bins does not affect the correctness of the protocol, but it does affect the computational overhead. Compared with CLR17[19], the strategy we adopt emphasizes that the size of the sets will change in real time in actual applications.

Suppose that receiver and sender insert their sets into respectively two hash tables with m' bins using an identical simple hash function. As a result, items with the same hash value are padded into the same bin and we only perform PSI protocol in each bin instead of whole sets.

Cuckoo hashing[25] is a one of ways to establish a hash table. In recent years, there are many works[26] using Cuckoo hashing in PSI. Compared with simple hashing mentioned above, Cuckoo hashing is proposed to solve the problem of hash conflict by using a little computation in exchange for better space utilization. We firstly consider the approach of simple hashing. Supposing there is a simple hash function $h : \{0,1\}^\alpha \rightarrow \{0,1\}^{\log m'}$, and m' is a power of 2, the conflict probability is $(m' - a)/m'$ when there are a empty bins. From the following specific strategy, we can learn how Cuckoo hashing contributes to dense hash table compared with simple hashing.

Permutation-based hashing[27] is suggested in [19] for the reason of storage advantage. We briefly recall this strategy using the notation of [19]. Receiver and sender parse each item from their sets as $x_L || x_R$, where the length of x_R is equal to $\log m'$. They insert (x_L, i) into hash table, where use $\mathtt{Hash}_i(x) = h_i(x_L) \oplus x_R$ to determine which bin to put in, $i \in [3]$. That means the lengths of each string stored in $\mathfrak{T}_X$ and $\mathfrak{T}_Y$ is reduced by $\sigma - \log m'$ bits from original stored (x, i). Arbitman et al. [19, 27] has already confirmed that $i = j$ and $x_L = y_L$ if $(x_L, i) = (y_L, j)$, so the correctness holds when applied permutation-based hashing.

The strategy of constructing hash table we used is similar to [19]. There are three independent random hash functions $h_1, h_2, h_3 : \{0,1\}^\alpha \rightarrow \{0,1\}^{\log m'}$, m' is a power of 2. The receiver perform original Cuckoo hashing while the sender perform another strategy, which is described in TABLE 1.

Input: Receiver inputs set Y and sender inputs set X, whose sizes are s_y and s_x, respectively. σ is public parameter.
output: Receiver outputs hash table $\mathfrak{T}_Y$ or $\perp$ while sender outputs hash table $\mathfrak{T}_X$.

- Sender: Let $\mathfrak{T}_X$ be an array of m' bins, each bin is initialized with $(\perp,\perp)$. For a item $x \in X$, sender selects smallest positive integer j s.t. $\mathfrak{T}_X[\texttt{Hash}_i(x)][j] = (\perp,\perp)$ or $\varnothing$ for all $i \in [3]$, and sets $\mathfrak{T}_X[\texttt{Hash}_i(x)][j] := (x_L, i)$. Otherwise, let $j = j+1$. Finally, sender outputs $\mathfrak{T}_X$.
- Receiver: Let $\mathfrak{T}_Y$ be an array of m bins ($m = \lceil \log s_y \rceil \le m'$), each bin is initialized with the value $(\perp,\perp)$. Receiver insert each item y in the following steps:
  (a) sets $w = y_L$ and $i \leftarrow [3]$;
  (b) defines and calls the function $\texttt{Insert}(w,j)$ as follows: swap (w,i) with the entry at the first m bits of $\mathfrak{T}_Y[\texttt{Hash}_i(w)]$. If $(w,i) \neq (\perp,\perp)$, recursively calls $\texttt{Insert}(w,j)$ where $j \leftarrow [3]\backslash\{i\}$.

  If receiver recursive calls $\texttt{Insert}$ exceeds $T = O(m)$ times for arbitrary y, let $m = m+1$ and perform hashing to bins again. Finally receiver outputs $\perp$ if $m > m'$, otherwise, outputs $\mathfrak{T}_Y$.

TABLE I
THE STRATEGY OF DYNAMIC HASHING TO BINS

According to receiver's strategy, we can infer that all bins in receiver's hash table are limited to at most one item since it is the most efficient option. If the receiver pad too many items in each bin, each item will do more comparisons for the number of items in the sender's bin will also increase. What's more, in order to ensure that the sender does not know which bin is empty, the receiver needs to pad dummy items in the empty bins. In fact, simple hashing works effectively if it's not required to encrypt receiver's dummy items.

The crucial difference from CLR17[19] is that we consider the establishment of the hash table under practical application. The sender's data set changes in real time, and m' should be set during the system initialization phase. Different receivers (customers) can choose the appropriate m as number of bins according to their size. Since m depends on the first m bits of $\texttt{Hash}$, so that the sender perform the PSI between the ciphertext of $\mathfrak{T}_Y[i]$ and $\mathfrak{T}_X[(i-1)(\log(m'/m)+1)+1 : i(\log(m'/m)+1)]$.

IV. Full PSI protocol based on TFHE and analysis

In this section, we detail the formal protocol in Algorithm 2, and then we analyze the correctness, the choice of actual parameters, efficiency and security.

A. Full PSI protocol based on TFHE

We extend our full protocol to arbitrary length of item while only small item of length σ can be support in the basic model and the optimization III-B is applied to increase efficiency.

The strategy of *dynamic hashing to bins* in III-B allows the receiver and the sender to set up their own hash table. In order to support larger length of items, hashing to shorter strings is necessary. Let $H' : \{0,1\}^* \rightarrow \{0,1\}^\omega$ (eg. $\omega = 128, 256$) be a random hash function. Both two parties perform $H'(X)$ and $H'(Y)$ respectively to shorten original sets if each item is oversize.

Since our encryption scheme encrypts the message to the exponent, it is impossible to encrypt an η-bits($\eta = \omega - \log m'$ for the reason of permutation based hashing in III-B) message x. We use a method called *partitioning*(different from [19]) to divide it into multiple substrings $x = x_1||x_2||\cdots||x_{\lceil \eta/\log n \rceil}$, $x_i \in \{0,1\}^n$ for each of i(add some 0 at the end of $x_{\lceil \eta/\log n \rceil}$-th string when it has less than n bits). Both receiver and sender perform *partitioning* after perform their respective hashing to bins strategy.

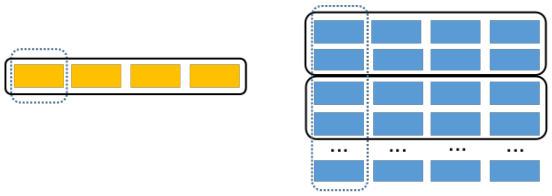
Fig. 1. Sketch Map of *Partitioning*.

Recalling to basic protocol in III-A, only the dotted line in the Fig.1 is considered. The information about Fig.1 is interpreted as follow: The yellow box on the left represents to one of the bins in the hash table of receiver, while the blue one refers to the corresponding bin of sender. Each line consists of $\lceil \eta/n \rceil = 4$ colored rectangle represents an item, which is divided into substrings. In the basic PSI protocol, only the dotted line is considered. In full PSI protocol, each substrings belongs to receiver must be compared with the corresponding column on the right. We compute the result of comparison through a series of AND gate.

It is worth noting that this method may cause failure. For an item of receiver, it may be considered to be included in the situation of that we have know from God' s perspective that it is not equal to any of the sender' s items. Therefore, we divide horizontally and use the OR gate to greatly reduce the probability of failure. The final failure probability is negligible, which is explained in detail in IV-B.

Now we describe the full PSI protocol in Algorithm 2 based on TFHE homomorphic encryption.

B. Efficiency and Security analysis

We set the following parameters for a typical scene, just supposing that item length is larger than 10^4 bits.

Algorithm 2 Full private set intersection ptotocol based on TFHE homomorphic encryption

Input: Receiver inputs set Y and sender inputs set X with size of s_y and s_x, respectively. s_y, s_x, σ and λ are public parameters.
output: Receiver outputs $X \cap Y$; sender outputs $\perp$.

1) **Setup**: Receiver and sender run TFHE.Setup(1^λ) to obtain public *paras*. Receiver keeps its secret key $\mathbf{s}$ and s.
2) **Preprocessing**: Sender and receiver hash their items to shorter strings, and then perform their respective hashing to bins strategy.
   (a) **Hashing to shorter strings**: If $2^{(\sigma-1)} > n$, then both parties hash their items to smaller representation. For a random hash function $H' : \{0,1\}^* \to \{0,1\}^\omega$. Perform the following steps with $(X, Y, \sigma) = (H'(X), H'(Y)), \omega)$. Otherwise, they do 2)(b) and turn to run Algorithm 1(omit some steps).
   (b) **Dynamic Hashing to bins**: Receiver perform Cuckoo hashing to obtain $\mathfrak{T}_Y$ and sender hash X into $\mathfrak{T}_X$ according to section III-B.
   (c) **Partitioning**: Sender and receiver perform *partitioning*, and then obtain $\{x_{i,j} | i \in m, j \in [\lceil \eta/\log n \rceil]\}$ and $\{y_{i,j} | i \in m', j \in [\lceil \eta/\log n \rceil]\}$ for $\eta = \sigma - \log m'$. If the bin is $(\perp, \perp)$, replace it by an illegal item.
3) **Encryption**: Receiver firstly encrypts each negative element $-y_{i,j}$ from $\mathfrak{T}_Y$ in the exponent (of x, e.g run RLWE.Encrypt(s, x^{-9}) when encrypted item is equal to 9), and then sends the m' RLWE ciphertexts ($\mathtt{ct}_1$, $\mathtt{ct}_2$, $\cdots$, $\mathtt{ct}_{m'}$) to the sender.
4) **Intersection**: For each $\mathtt{ct}_g$, sender
   (a) select $\mathfrak{T}_X[(k-1)(\log(m'/m)+1)+1 : k(\log(m'/m)+1)]$ denoted by $\{(x_{i,j})\}$ for $i \in \gamma, j \in \lceil \eta/\log n \rceil$ where γ is number of items in this sub table.
   (b) partitions its set into $\lceil \gamma/l \rceil$ subsets horizontally for l is the number of sender's substrings concluded in one $p(x)$ (refer to Fig.1).
   (c) let $p_{k,j}(x) = x^{x_{(k-1)\eta,j}} + x^{x_{(k-1)\eta+1,j}} + \cdots + x^{x_{k\eta-1,j}}$, and compute $\vec{c}_{k,j} = \text{Extract}(p_{k,j}(x) \cdot \mathtt{ct}_g)$ for $k \in \lceil \gamma/l \rceil$ and $j \in \lceil \eta/\log n \rceil$.
   (d) do following procedure, and return $\vec{p}_g = \vec{c}_{\lceil \gamma/l \rceil+1, \lceil \eta/\log n \rceil+1}$.

```
for $k = 1$ to $\lceil \gamma/l \rceil - 1$ do
    for $j = 1$ to $\lceil \eta/\log n \rceil - 1$ do
        $\vec{c}_{k,j+2} = \text{HomAND}(\vec{c}_{k,j}, \vec{c}_{k,j+1})$
    end for
    $\vec{c}_{\lceil \gamma/l \rceil+1, \lceil \eta/\log n \rceil+1} = \text{HomOR}(\vec{c}_{k,\lceil \eta/\log n \rceil+1}, \vec{c}_{k+1,\lceil \eta/\log n \rceil+1})$
end for
```

   Sender return m' LWE ciphertexts that writing in $(\vec{p}_1, \vec{p}_2, \cdots, \vec{p}_{m'})$ to receiver.
5) **Decryption**: Receiver decrypts every LWE ciphertext and outputs $\{y_i : \text{FHE.Decrypt}(\mathbf{s}, \vec{p}_i) = 1\}$.

$\lvert X \rvert$	$\lvert Y \rvert$	n	m'	m
10^5	100	1024	1024	128
10^5	23	1024	1024	32

For $|Y| = 100$ in the first row of the table above, we set other parameters as $\eta = 118, l = 100$. Sender perform around $3|X|(\lceil \eta/\log n \rceil - 1)/(lm) = 257$ gates for each item. Receiver sends $m = 128$ RLWE ciphertexts and receive $m = 128$ LWE ciphertexts.

Failure Probability Recalling to Fig.1 and the interpretation from IV-A, we know that there may occur a failure that each item from receiver may be mistakenly considered to be included in the sender's set. Using the notation of Algorithm 2, we can calculate the probability of failure as around $(l/n)^{-\lceil \eta/\log n \rceil}$, with a precise example is around 2^{-41} for very conservative parameters $l = 100, n = 1024, \eta = 118$.

Semi-honest Security of Full Protocol We present the simulation-based security in the semi-honest setting. The formal definition of the standard semi-honest simulation based security refers to [12]. The definition claims that if a party's view during a protocol execution is simulatable when given its input and output, then the party learns nothing from the protocol execution beyond what they can derive from their input and prescribed output. Readers can find more details in [12].

We prove our security by illustrating the existence of simulators of two parties, respectively, which are defined in the next paragraph. We assume that the parameters $(n, N, h_1, h_2, h_3, m', H')$ are declared as public input at the beginning of the protocol and we present the simulator of sender and receiver as Sim_S and Sim_R respectively. Following the definition in [12], Sim_R holds the input of set Y while Sim_S can only access to X.

The proof of the corrupt sender is straightforward depending on the IND-CPA security of the FHE scheme we based on. The sender ' s simulator Sim_S encrypt zero using fully homomorphic encryption. Then these ciphertexts are indistinguishable from the sender ' s view in the real protocol.

From the perspective of the receiver, Sim_R replace items belongs to $X \cap Y$ to zero and randomly select non-zero elements in Z_t to fill in the vacant slots in the hash table with m bins by the strategy of Cuckoo hashing. Then the Sim_R splits each bin into $\lceil \eta / \log n \rceil \lceil \gamma / l \rceil$ parts to obtain $\lceil \eta / \log n \rceil \lceil \gamma / l \rceil$ subsets. We put these slots in each subsets on the exponent to get $m' \lceil \eta / \log n \rceil \lceil \gamma / l \rceil$ polynomials. Homomorphically encrypt the polynomials, then we can finally achieve a simulation of the receiver ' s view according to the circuit security of the underlying fully homomorphic encryption.

Attempts and Barriers to Obtain Malicious security. From the receiver's perspective, the ability that a malicious receiver can fill some extra queries in the empty bins to achieve additional information becomes a crucial barrier to obtain malicious security as in [19]. Moreover, the problem of circuit privacy described in [19] is also occurs to our protocol but can be ignored when applied to practical applications.

In the setting that we give almost computation workload to the sender instead of let two parties interact their intermediate results, the corrupt behavior of the sender is inevitable. An obvious example is that a malicious sender can send back encryption of constant 1, then the receiver believes that the intersection equals to the whole set Y. There is no efficient way to avoid such behavior even combining with techniques such as OPRF. Although OPRF can be used to achieve a security of privacy against malicious sender[20], it still cause two more rounds of interaction, which is exactly what we want to avoid in our protocol.

V. Conclusion

We propose a PSI protocol based on TFHE homomorphic encryption, and the bootstrapping is also concluded to ensure the correctness of computation. The significance of our work lies not only in the PSI protocol itself, but also in providing applications of FHE, especially the latest TFHE scheme. As far as we know, it has fewer applications compared with leveled FHE such as FV/BGV[15, 16] scheme.

In addition, TFHE supports arbitrary depth of homomorphic operations, which avoids those optimizations[19] made to reduce the depth of the circuit, resulting in additional computation and communication complexity increased to $O(|Y| + \log |X|)$. Our protocol communication complexity only depends on the set size of receiver, which is $O(|Y|)$. Unfortunately, TFHE does not have efficient batching technique, which can greatly increases the efficient of communication and computation. Lacking of batching technique especially limits the practical application when the receiver ' set is relatively large. We are also looking forward to TFHE's batching technique and add it to our PSI protocol.

Finally, we also look forward to further discussions about PSI protocol based on fully homomorphic encryption, and we will implement our protocol using TFHE ++ library and analysis its efficiency more detailed according to the implementation in the future.

References

[1] Benny Pinkas, Thomas Schneider, Gil Segev, and Michael Zohner. Phasing: Private set intersection using permutation-based hashing. In 24th USENIX Security Symposium, USENIX Security 15, Washington, D.C., USA, August 12-14, 2015., pages 515–530, 2015.

[2] Benny Pinkas, Thomas Schneider, and Michael Zohner. Scalable private set intersection based on OT extension. Cryptology ePrint Archive, Report 2016/930, 2016. http://eprint.iacr.org/2016/930.

[3] Michele Orr'u, Emmanuela Orsini, and Peter Scholl. Actively secure 1-out-of-n OT extension with application to private set intersection. In Topics in Cryptology - CT - RSA 2017 - The Cryptographers' Track at the RSA Conference 2017, San Francisco, CA, USA, February 14-17, 2017, Proceedings, pages 381–396, 2017.

[4] Vladimir Kolesnikov, Ranjit Kumaresan, Mike Rosulek, and Ni Trieu. Effiffifficient batched oblivious PRF with applications to private set intersection. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, ACM CCS 16, pages 818–829. ACM Press, October 2016.

[5] Peter Rindal and Mike Rosulek. Improved private set intersection against malicious adversaries. In Jean-S´ebastien Coron and Jesper Buus Nielsen, editors, EUROCRYPT 2017, Part I, volume 10210 of LNCS, pages 235–259. Springer, Heidelberg, May 2017.

[6] Benny Pinkas, Mike Rosulek, Ni Trieu, and Avishay Yanai. Spot-light: Lightweight private set intersection from sparse ot extension. In CRYPTO, 2019.

[7] Carmit Hazay. Oblivious polynomial evaluation and secure set-intersection from algebraic PRFs. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, TCC 2015, Part II, volume 9015 of LNCS, pages 90–120. Springer, Heidelberg, March 2015.

[8] Michael J. Freedman, Carmit Hazay, Kobbi Nissim, and Benny Pinkas. Efficient set intersection with simulation-based security. Journal of Cryptology, 29(1):115–155, January 2016.

[9] Agnes Kiss, Jian Liu, Thomas Schneider, N. Asokan, and Benny Pinkas. Private set intersection for unequal set sizes with mobile applications. PoPETs, 2017(4):177–197, 2017.

[10] Benny Pinkas, Thomas Schneider, Oleksandr Tkachenko, and Avishay Yanai. Efficient circuit-based PSI with linear communication. In EUROCRYPT, pages 122–153, 2019.

[11] Léo Ducas and Damien Stehlé. Sanitization of FHE ciphertexts. In Annual International Conference on the Theory and Applications of Cryptographic Techniques, pages 294–310. Springer, 2016.

[12] Yehuda Lindell. How to simulate it - a tutorial on the simulation proof technique. Cryptology ePrint Archive, Report 2016/046, 2016. http://eprint.iacr.org/2016/046.

[13] Craig Gentry, Fully homomorphic encryption using ideal lattices, Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009, 2009, pp. 169–178.

[14] R L Rivest, L Adleman, and M L Dertouzos, On data banks and privacy homomorphisms, Foundations of Secure Computation, Academia Press (1978), 169–179.

[15] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan, (Leveled) fully homomorphic encryption without bootstrapping, ACM Trans. Comput. Theory 6 (2014), no. 3, Art. 13, 36. MR 3255281.

[16] J. Fan and F. Vercauteren. Somewhat practical fully homomorphic encryption. IACR Cryptology ePrint Archive, 2012:144, 2012.

[17] J. H. Cheon, A. Kim, M. Kim, and Y. Song. Homomorphic encryption for arithmetic of approximate numbers. In Advances in Cryptology–ASIACRYPT 2017: 23rd International Conference on the Theory and Application of Cryptology and Information Security, pages 409–437. Springer, 2017.

[18] Cheon, J.H., Han, K., Kim, A., Kim, M., Song, Y.: Bootstrapping for approximate homomorphic encryption. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 360–384. Springer (2018).

[19] Chen, H., Laine, K., Rindal, P.: Fast private set intersection from homomorphic encryption. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. pp. 1243–1255. ACM (2017).

[20] Hao Chen, Zhicong Huang, Kim Laine, and Peter Rindal: Labeled PSI from Fully Homomorphic Encryption with Malicious Security. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. ACM (2018).

[21] Florian Bourse, Olivier Sanders, Jacques Traoré. Improved Secure Integer Comparison via Homomorphic Encryption. In CT-RSA 2020, pages 391-416. Springer, San Francisco, February 2020.

[22] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabach‘ene. Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. In Jung Hee Cheon and Tsuyoshi Takagi, editors, ASIACRYPT 2016, Part I, volume 10031 of LNCS, pages 3–33. Springer, Heidelberg, December 2016.

[23] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabach‘ene. Faster packed homomorphic operations and efficient circuit bootstrapping for TFHE. In Tsuyoshi Takagi and Thomas Peyrin, editors, ASIACRYPT 2017, Part I, volume 10624 of LNCS, pages 377–408. Springer, Heidelberg, December 2017.

[24] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabach‘ene. TFHE: Fast fully homomorphic encryption over the torus. Cryptology ePrint Archive, Report 2018/421, 2018. https://eprint.iacr.org/2018/421.

[25] Rasmus Pagh and Flemming Friche Rodler. Cuckoo hashing. In European Symposium on Algorithms, pages 121–133. Springer, 2001.

[26] Benny Pinkas, Thomas Schneider, Christian Weinert, and Udi Wieder. Efficient circuit-based PSI via cuckoo hashing. In EUROCRYPT, 2018.

[27] Yuriy Arbitman, Moni Naor, and Gil Segev. Backyard cuckoo hashing: Constant worst-case operations with a succinct representation. In Foundations of Computer Science (FOCS), 2010 51st Annual IEEE Symposium on, pages 787–796. IEEE, 2010.

[28] Hazay, C., Lindell, Y.: Efficient protocols for set intersection and pattern matching with security against malicious and covert adversaries. In: Canetti, R. (ed.) Theory of Cryptography. pp. 155–175. Springer Berlin Heidelberg, Berlin, Heidelberg (2008).