Nearly Optimal Protocols for Computing Multi-party Private Set Union

Xuhui Gong, Qiang-Sheng Hua*, Hai Jin
National Engineering Research Center–Big Data Technology and System Lab,
Key Laboratory of Services Computing Technology and System, Key Laboratory of Cluster and Grid Computing,
School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China.

2022 IEEE/ACM 30th International Symposium on Quality of Service (IWQoS) | 978-1-6654-6824-4/22/$31.00 ©2022 IEEE | DOI: 10.1109/IWQoS54832.2022.9812897

***Abstract*—Private Set Operations (PSO) are a hot research topic and one of the most extensive research problems in data mining. In the PSO, Multi-party Private Set Union (MPSU) is one of the fundamental problems. It allows some participants to learn the union of their data sets without leaking any useful information. However, most of the existing works have high communication, computation and round complexities. In this paper, we first propose a novel and efficient protocol to securely compute MPSU under the semi-honest model. In our system model, there exist n participants where each participant has a set of size k (k could be different among participants). There are also up to t $(0 \le t < n)$ participants which could collude with each other. We suppose the communication channels among participants are insecure and can easily suffer from eavesdropping attacks. Our first protocol using element computing algorithm and Homomorphic Encryption, i.e., HE-MPSU, only requires $O(1)$ rounds and has $O(nN\lambda)$ communication complexity which almost matches the communication lower bound $\Omega(nN/\log n)$ for the MPSU problem, where λ is a security parameter and N ($k \le N \le nk$) is the set union cardinality. In addition, we note that for the two-party case, i.e., $n = 2$, our HE-MPSU protocol has the same complexities as the state-of-the-art work in [1]. For this special case, i.e., two-party Private Set Union (PSU), we further optimize and design a more efficient protocol using oblivious transfer (OT) protocol, i.e., OT-PSU. It only requires $O(1)$ rounds and $O(k\lambda)$ communication complexity which almost matches the communication lower bound $\Omega(k)$. More importantly, it avoids using computationally expensive public-key operations (exponentiations). In other words, the number of exponentiations in this protocol is independent of the size of the data sets. Compared with the existing protocols, our two protocols have the lowest communication, computation and round complexities.**

***Index Terms*—Set union, Bloom Filter, homomorphic encryption, multi-party secure computation**

I. Introduction

Private set operations (PSO) deal with a class of problems by using sensitive data of distributed participants. They contain private set intersection [2], [3], private set union [4], private set union/intersection cardinality (PSU-CA/PSI-CA) [5], and so on. These problems have many important application scenarios and efficient solutions are highly desirable. Here is a simple example for MPSU [6], [7]. In cyber risk assessment and management, organizations want to optimize their security updates and to minimize vulnerabilities. The usual method is to compute a joint list of their blacklisted IP addresses and other relevant data information. While they are unwilling to expose some sensitive network data. Furthermore, PSO has been used in the privacy-preserving graph algorithm [4] and data mining [8].

This line of research has drawn a significant amount of attention in the community and several relevant techniques to solve these problems have been proposed. Secure Multi-party Computation [9], [10] can be used to solve PSO [4]. However, it usually incurs too high computation overhead to apply to practical applications. Homomorphic Encryption (HE) techniques can be applied to tackle PSO. It can allow addition or multiplication operations directly on ciphertexts without decrypting operations. Freedman et al. [11] utilize oblivious polynomial evaluation (OPE) and HE techniques to solve PSI. In [2], [12], [13], these authors introduced adaptable ways to compute the multi-party PSO or PSU based on similar methods. However, their works still require high computation overhead.

In this paper, we revisit the classic MPSU. There exist n participants who want to compute the union of all participants' private datasets. Meanwhile, the process does not reveal additional information other than what can be obtained from the set union. We propose two privacy-preserving and more efficient protocols with low communication, computation and round complexities, to solve MPSU and PSU, respectively. In our HE-MPSU protocol, we design a novel element computing algorithm and construct a global Bloom Filter. A brief comparison of the complexities between previous protocols and our protocols is shown in Table I.

We make the following contributions:

- We give two efficient protocols, i.e., HE-MPSU in the multi-party and OT-PSU in the two-party, which can obtain the set union with probability at least $1-\delta$, where $0 < \delta < 1$.
- Our HE-MPSU protocol can resist up to t collusive participants which collude with each other and try to learn more privacy information of other participants.
- The HE-MPSU protocol only requires $O(1)$ rounds.
- The HE-MPSU and OT-PSU protocols almost match the lower bound $\Omega(nN/\log n)$ and $\Omega(k)$ in terms of communication complexity, respectively.
- The number of exponentiations of the OT-PSU protocol does not depend on the size of the datasets.

* Qiang-Sheng Hua (qshua@hust.edu.cn) is the corresponding author.

978-1-6654-6824-4/22/$31.00 ©2022 IEEE

The remainder of our paper is organized as follows. Section II briefly discusses related works. Section III outlines the system model, problem definition, security model, design goals and related techniques. Our protocols' overview is presented in section IV. The HE-MPSU and OT-PSU protocols are given in Sections V and VI, respectively. Section VII gives more detailed analyses for the correctness, security and complexities of the two protocols. Section VIII discusses how to apply our algorithms to the case where the set sizes are different for each participant. We summarize our work in Section IX.

II. Related Work

Private set operations are a very hot research topic. They contain private set union, private set intersection, private set union or intersection cardinality, and so on. Kissner et al. [2] and Sang et al. [13] have presented several multiparty protocols for computing PSO including the PSU and PSU-CA, based on OPE and additively homomorphic encryption (AHE). A similar idea is used by Frikken in [12]. In [3], multi-party PSO protocols were proposed by using generic secure multi-party computation. Their schemes can also compute MPSU making use of oblivious sorting and basic comparison operations. In [16], an MPSU protocol with constant-round was proposed by using the secret-sharing scheme and Reversed Laurent Series. However, they consume high communication overhead and computation costs.

There exist some works which were proposed by relaxing privacy requirements. The protocol in [8] can reveal superfluous information, such as set intersection cardinality among participants. Many et al. [18] proposed an MPSU protocol based on Bloom Filters, but it needs to traverse the entire data space for computing set union. Shishido et al. [19] presented an MPSU protocol for multisets based on Bloom Filters. However, their protocol needs to traverse the entire data space for computing set union as well and can expose the multiplicities of all elements in the set union. Two works [20] and [5] study the approximate PSU-CA and cannot be used for computing PSU. In [15], the authors proposed new protocols for computing private set operations by using the vector encoding method. But the computation and communication complexities of their protocols are related to the size of the entire data space. Note that, in [21], [22], the authors presented privacy-preserving data aggregation protocols which can compute arbitrary functions, including the set union. But their schemes can expose the multiplicity of each element in the set union.

Additionally, there exist many works for two-party protocols. Brickell et al. in [4] have proposed two PSU schemes based on the iterative and the tree-pruning methods. Hazay et al. [23] utilized oblivious pesudorandom function evaluation and ElGamal encryption under the malicious model. Davidson et al. have proposed PSU/PSI protocols based on Bloom Filter and AHE in [1], [24]. In [7], a new efficient protocol for computing PSU is designed based on oblivious transfer techniques. However, these protocols have high communication and computation costs.

III. Preliminaries

In the section, we introduce our system model, problem definition, security model, design goals and some related techniques used in our paper.

A. System Model

We consider a classical model which contains n participants $p_0, p_1, ..., p_{n-1}$. Each p_j $(j \in \{0,...,n-1\})$ holds a private data set $S_j = \{x_{j,1}, ..., x_{j,k}\}$ of size k. Note that, for simplicity, we suppose that the private data sets of all participants are equal in size, i.e., $|S_j| = k$ for all $j \in \{0,1,...,n-1\}$. We will discuss how to extend our algorithms to the case where the sizes of each participant's data set are different in Section VIII. There exist bi-direction communication channels between participants. For any positive integer a, we use $[a]$ and $[1,a]$ to denote the sets $\{0,1,...,a-1\}$ and $\{1,...,a\}$, respectively. W.l.o.g., we suppose S_i is obtained from the universal set $[1,M]$, i.e., $S_i \subseteq [1,M]$, for all $i \in [n]$, where M is a positive integer. Let $l = \lceil \log M \rceil$ and $y \in_R Y$ represent the element y randomly sampled from the set Y. We denote the set union $\cup_{j\in[n]} S_j$ by $\mathscr{U}$. The notations are summarized in Table II.

B. Problem Definition

All participants want to compute the set union $\mathscr{U}$ based on their own private sets and do not reveal additional privacy information (other than what can be obtained from the set union).

Now, we formally introduce the multi-party private set union definition as follows.

Definition 1. *In the MPSU problem, participants only obtain the set union $\mathscr{U} = S_0 \cup S_1 \cup \cdots \cup S_{n-1} = \{\beta_1, ..., \beta_N\}$ without obtaining extra knowledge (other than what can be obtained from the set union), where $N = |S_0 \cup S_1 \cup \cdots \cup S_{n-1}|$, $\beta_i \neq \beta_j$ and β_i, $\beta_j \in S_0 \cup S_1 \cup \cdots \cup S_{n-1}$ for any $i \neq j \in [1,N]$.*

C. Security Model

We focus on the semi-honest model. That is, participants faithfully perform a protocol and are interested in obtaining additional information of other participants. Furthermore, our multi-party protocol can tolerate up to t collusive participants which could share their own private data with each other and try to get more privacy information of other participants.

To prove the security for a protocol, we employ the standard simulation paradigm [25]. Let $\mathbb{M}$ denote an MPSU protocol. We use $view_i$ $(i \in [n])$ to represent the view of the participant p_i, respectively. $view_i$ includes $(S_i, r_i, m_{1,i}, ..., m_{j,i})$, where S_i is the private data set of participant p_i, r_i is the result of p_i's internal random coin tosses and $m_{j,i}$ is the j-th message p_i received. For $I = \{i_1, ..., i_s\} \subseteq \{0,1,...,n-1\}$ $(s \leq t)$, let P_I, X_I, $view_I$ denote the participants $\{p_{i_1}, ..., p_{i_s}\}$, $\{S_{i_1}, ..., S_{i_s}\}$, the view of the s collusive participants, respectively. Let $\overset{c}{\equiv}$ denote computational indistinguishability.

Definition 2. *A protocol $\mathbb{M}$ can securely compute the MPSU problem and can tolerate up to t colluding participants P_I, if*

TABLE I
RESULT COMPARISONS FOR MPSU AND PSU

Communication	Computation	Threshold	Round	Protocol	Multi-party?	Communication Complexity Lower Bound
$O(n^2k\lambda)$	$O(n^2k^2)^{\ddagger}$	$t<n$	$O(n)$	[12]	Y	$\Omega(nN/\log n)^{+}$ [14]
$O(nM\lambda)$	$O(nM)^{\ddagger}$	$t<n$	$O(n)$	[15]	Y	
$O\left(n^3k^2\lambda\right)$	$O\left(n^4k^2\right)^{*}$	$t<n/2$	$O(1)$	[16]	Y	
$O(nN\lambda)$	$O(nN)^{\ddagger}$	$t<n$	$O(1)$	HE-MPSU	Y	
$O(k\lambda\log k)$	$O(k\log k)^{*}$	–	$O(1)$	[7]	N	$\Omega(k)$ [11]
$O(k\lambda\log k)$	$O(k\log k)^{*}$	–	$O(1)$	[17]	N	
$O(k\lambda)$	$O(k)^{\ddagger}$	–	$O(1)$	[1]	N	
$O(k\lambda)$	$O(\lambda)^{\ddagger}$	–	$O(1)$	OT-PSU	N	

λ represents the security parameter. *The upper bounds mean the number of modular multiplication operations. ‡These upper bounds denote the number of modular exponentiation operations. $^{+}$This is the communication lower bound for exactly computing multi-party PSU-CA. However, since $|S_0\cup S_1\cup\cdots\cup S_{n-1}|=N$, it is also the communication lower bound for computing MPSU.

TABLE II
NOTATIONS

Symbols	Descriptions
n, t	Total number of participants, number of collusive participants
p_j, S_j	j-th participant, private data set of participant p_j
$[n]$, λ	Set $\{0,1,2,...,n-1\}$, security parameter
k	Size of each p_j's private data set, i.e., $\lvert S_j\rvert$, for all $j\in[n]$
$y\in_R Y$	The element y randomly sampled from the set Y
$\mathscr{U}$, N	The set union $\bigcup_{j\in[n]}S_j$, cardinality of the set union $\mathscr{U}$
$\overset{c}{\equiv}$	Computational indistinguishability
H	Hash functions $\{h_0(\cdot),h_1(\cdot),...,h_{k_1-1}(\cdot)\}$
$[t+1]$, $[1,m]$	Set $\{0,...,t\}$, set $\{1,...,m\}$
p, q	Two large primes
$\overline{n}$	The product of the two large primes p and q, i.e., $\overline{n}=pq$

there exists a probabilistic polynomial-time (PPT) simulator S such that

$$S\left(P_I,X_I,\mathbb{M}(S_0,...S_{n-1})\right)\overset{c}{\equiv} view_I^{\pi}(S_0,...S_{n-1}). \tag{1}$$

D. Design Goals

1) ***Correctness*:** The proposed protocols should generate the set union with probability at least $1-\delta$, where $0<\delta<1$.
2) ***Security*:** The proposed protocols should satisfy the security model, i.e., Definition 2. No participants can learn any extra knowledge other than what can be obtained from the set union. In other words, the protocol mainly focuses on two aspects for security. One is that it needs to hide the multiplicities of all elements in the set union. The other is that it preserves the information of which element comes from which participant.
3) ***Efficiency*:** The proposed protocols should have a low communication complexity (total number of sent bits), low computation complexity (total number of computation operations) and low round complexity (a round means a phase in which all participants can simultaneously exchange messages).

E. FM Sketches

First, let's briefly recall FM sketches presented in [26]. They are used to estimate the cardinality of a multiset S. FM sketch is built on an l-bit binary vector $(b_0,...,b_{l-1})$. Meanwhile a function $\overline{h}(\cdot)$ is used. For any element y, the probability that $\overline{h}(y)=i$ $(i\in[l]=\{0,..,l-1\})$ is equal to $\Pr[\overline{h}(y)=i]=2^{-(i+1)}$ (the function $\overline{h}(\cdot)$ can be constructed from a uniform hash function by counting the number of trailing zeroes). Initially, all bits in the vector $(b_0,...,b_{l-1})$ are set to zero. For each element y of the multiset S, anyone uses $\overline{h}(\cdot)$ to compute $\overline{h}(y)$ and sets $b_{\overline{h}(y)}=1$. By using the smallest index z which meets $b_z=0$, we can estimate $|S|$ by $\frac{2^z}{\phi}$, where $\phi=0.77351$. For the standard deviation, it could be reduced by using m_0 sketches in parallel and get m_0 estimators $z^{\langle 1\rangle},\ldots,z^{\langle m_0\rangle}$. Then, we utilize $\frac{Z}{m_0}$ to estimate $|S|$ and the standard deviation is reduced to $\frac{1.12}{\sqrt{m_0}}$, where $Z=z^{\langle 1\rangle}+\ldots+z^{\langle m_0\rangle}$. To improve accuracy, the authors in [27] give a modified formula as follows,

$$\widetilde{N}=\frac{2^{\frac{Z}{m_0}}-2^{-\kappa\cdot\frac{Z}{m_0}}}{\phi},\text{where }\kappa=1.75. \tag{2}$$

According to the equation, the authors in [20] give the below lemma.

Lemma 1 ([20])**.** *For n sets $S_0,...,S_{n-1}$, let $\widetilde{N}$ be the estimation for $N=|S_0\cup S_1\cup\cdots\cup S_{n-1}|$ by using Equation (2). For any $\varepsilon,\delta_1\in(0,1)$, we have $\Pr\left[|\widetilde{N}-N|\le\varepsilon N\right]\ge 1-\delta_1$, where $m_0\ge 2.5088\cdot(\frac{\mathrm{erf}^{-1}(1-\delta_1)}{\min(-\log(1-\varepsilon),\log(1+\varepsilon))})^2$ and $\mathrm{erf}^{-1}(x)$ is the inverse error function.*

F. Bloom Filter

The Bloom Filter (BF) [28] is a probabilistic data structure. It can efficiently represent sets and allow for membership check. To be specific, it uses an m-bit array $BF=(b_1,...,b_m)$ to represent a set $Y=\{y_1,...,y_n\}$ of size n. Initially, each bit in the array BF is set 0 and k_1 independent uniform hash functions $H=\{h_i(\cdot)\mid i\in[k_1]\}$ map elements to range $[1,m]=\{1,...,m\}$. To represent any element $y\in Y$ in this filter, for all $i=[k_1]=\{0,1,...,k_1-1\}$, the bits $b_{h_i(y)}$ set to one, i.e., $b_{h_i(y)}=1$. To check an element y whether it belongs to Y, anyone only needs to check values of all bits $b_{h_i(y)}$ $(i\in[k_1])$. If all bits are 1, $y\in Y$ and $y\notin Y$ otherwise. Because k_1 hash functions may cause collisions, they introduce false positives, i.e., y does not belong to S but $b_{h_i(y)}=1$ for all $i=[k_1]$. The probability that any bit is set to 1 is $p_1=1-(1-1/m)^{k_1n}$ in the Bloom Filter. As shown in [29], the upper bound of the probability that the filter returns a false positive is $\varepsilon'=p_1^{k_1}\times\left(1+O(\frac{k_1}{p_1}\sqrt{(\ln m-k_1\ln p_1)/m})\right)$, which

is negligible for k_1. To decrease the false positive probability ε', k_1 is set to $k_1 = (m/n)\ln 2$ and $m \geq n\log_2(1/\varepsilon') \cdot \log_2 e$. We can set the optimal parameters $k_1 = \log_2(1/\varepsilon')$ and $m = n\log_2(1/\varepsilon') \cdot \log_2 e$ [29].

G. Cryptographic Building Blocks

In our protocols, we use an additively homomorphic encryption, such as Paillier cryptosystem [30]. It permits anyone to execute computations on its ciphertexts without decrypting it. To be precise, given the ciphertexts $\mathbf{Enc}(m_1)$ and $\mathbf{Enc}(m_2)$, anyone can compute $\mathbf{Enc}(m_1 + m_2) = \mathbf{Enc}(m_1) \cdot \mathbf{Enc}(m_2)$, where $\mathbf{Enc}(m_j)$ denotes the ciphertexts of the plaintext message m_j $(j \in \{1,2\})$, and $+$, $\cdot$ denote modular addition and modular multiplication operations on the plaintexts and ciphertexts, respectively. For convenience, we omit the modular multiplication operation symbol, i.e., $\mathbf{Enc}(m_1 + m_2) = \mathbf{Enc}(m_1)\mathbf{Enc}(m_2)$. Moreover, $(\mathbf{Enc}(m_1))^c = \mathbf{Enc}(m_1 \times c)$ for any constant c, where $(\mathbf{Enc}(m_1))^c$ denote modulo exponential operation on the ciphertext $\mathbf{Enc}(m_1)$, and $\times$ refers to modulo multiplication operation on the plaintexts. Similarly, we omit the modular multiplication operation symbol, i.e., $\mathbf{Enc}(m_1)^c = \mathbf{Enc}(cm_1)$. In addition, based on its additive homomorphism, we can re-encrypt encryption of m_1, i.e., $\mathbf{Enc}(m_1)$, to generate another encryption of m_1 by computing $\mathbf{Enc}(m_1) \leftarrow \mathbf{Enc}(m_1)\mathbf{Enc}(0) = \mathbf{Enc}(m_1 + 0) = \mathbf{Enc}(m_1)$ (we still refer to it as $\mathbf{Enc}(m_1)$).

In order to withstand t colluding participants, we adopt the (n,t)-threshold variant of Paillier cryptosystem [31]. In this scheme, each participant possesses a share of a private key for the cryptosystem and colluding participants cannot obtain additional information of the private key. For any ciphertext, it can only be decrypted when more than t participants perform decryption operations together. Similar to the idea of [32], we need a method that permits participants to determine whether or not $\mathbf{Enc}(m)$ is encryption of zero without revealing anything. This can be realized by randomizing the ciphertext and by jointly decrypting it.

Note that, in [33], the authors propose a probabilistic coding scheme for privacy-preserving Min computation based on XOR operation. Specifically, a participant has a status $r \in$ {"affirmative", "negative"}. If $r =$"affirmative", corresponding code $C(r) = 0^q$; otherwise, $C(r) \in_R \{0,1\}^q$. In this paper, a similar idea of the encryption technique is utilized. But their method is not used to solve MPSU directly. Because it can expose the multiplicity of each element in the set union.

Oblivious Transfer (OT) [34] [35] is an important primitive for secure computation which involves sender and receiver. The sender holds a λ-bit message pair (m_0, m_1) and the receiver has a bit b. As the result of OT, the receiver gets the message m_b and learns nothing about m_{1-b}. Meanwhile, the sender does not acquire any information for the bit b. Let $\mathrm{OT}_\lambda^{\mathrm{m}}$ denote that this type of OT is executed m invocations on m message pairs. Ishai et al. [36] present a practical OT extension so that lots of OT protocols can be constructed by using a few expensive OT protocols and cheap symmetric-key operations. Namely, the OT extension can reduce $\mathrm{OT}_\lambda^{\mathrm{m}}$ to $\mathrm{OT}_\lambda^{\lambda}$.

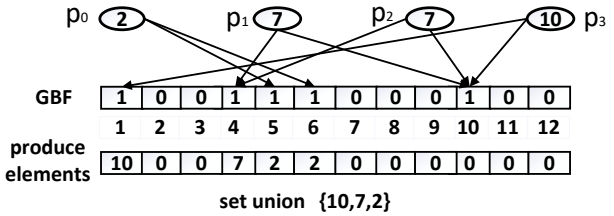

Fig. 1. An illustrating example of our protocols. There are 4 participants $p_0, ..., p_3$ which have elements 2,7,7,10, respectively. They construct a (Global) Bloom Filter (which has 12 bits) by using two hash functions $h_0(x)$ and $h_1(x)$ (we assume $h_0(x) = ((2x+1) \mod 12) + 1$ and $h_1(x) = ((x+2) \mod 12) + 1$). According to the BF, they produce elements and only save the elements that only this element is mapped to some bits in BF. In the BF, the 1-st, 4-th, 5-th, 6-th bit positions have unique elements mapped in. Thus we can obtain 10,7,2. But in the 10-th bit positions of BF, there are two elements 7 and 10 which are mapped in.

There have been many efficient OT extension protocols which have been proposed such as [37] [38].

IV. Protocols Overview

Our two protocols use the Bloom Filter to compute possible elements belonging to a set union. But the use of the Bloom Filter in our protocols is significantly different from the previous works [24], [1], [18], [5]. The idea of our protocols is based on a key observation which can be used to reduce communication and computation complexities. The key observation is that, for the Bloom Filter $BF = (b_1, ...b_m)$ produced by a set union, each element in the set union has at least one bit b_i $(i \in [1,m])$ where only this element is mapped to with probability at least $1 - \delta_3$ (see Lemma 2). Based on the proposed techniques and the key observation, we can compute all elements of set union with low complexities. In some sense, for each element in the set union, we only use one bit in the Bloom Filter. However, the previous works for each element use the Bloom Filter directly and need to use all bits which are mapped to by using k_1 hash functions. Thus, they are very suitable for two-party computing. But it does not apply in the multi-party case. Because it is difficult to restore all elements in the union set (needs to traverse data space). Most of the previous works rely on oblivious polynomial evaluation. The methods inherently need the rounds which are dependent on the number of participants. Fig. 1 shows an illustrating example for our protocols.

V. HE-MPSU

In this section, we show our HE-MPSU protocol which contains four stages: the system setup stage, the approximating stage, the global BF (GBF) constructing stage and the element computing stage. We show the overview of the four stages of our HE-MPSU protocol in Fig.2. In the following, we describe this four-stage process in detail.

A. The System Setup Stage

In order to protect participants' privacy, participants invoke the (n,t)-threshold Paillier cryptosystem [31] (more precisely, its key generation algorithm) based on the secure parameter λ,

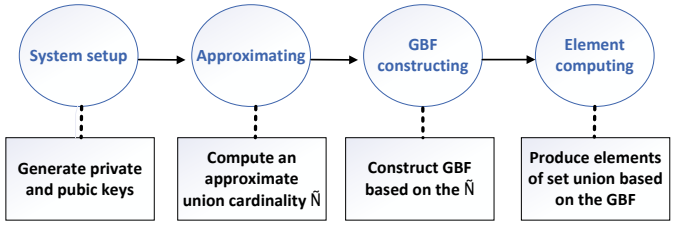

Fig. 2. The four stages overview of our HE-MPSU protocol

Algorithm 1: ORPE $(\{p_j, b_{j,*}, sk_j, pk\}_{j\in[n]})$

Input: The bit $b_{j,*}$, private key sk_j and public key pk of participant p_j $(j \in [n])$;
Output: Perturbed data e_*;

1 **for** *each* p_j $(j \in [n])$ **do**
2 If $b_{j,*} = 0$, p_j computes $\overline{b}_{j,*} = \mathbf{Enc}(0)$;
3 If $b_{j,*} = 1$, p_j computes $\overline{b}_{j,*} = \mathbf{Enc}(e_{j,*})$, where $e_{j,*} \in_R Z_{\overline{n}}$;
4 p_j sends $\overline{b}_{j,*}$ to p_0;
5 p_0 computes $C_* = \prod_{j=0}^{n} \overline{b}_{j,*}$ and sends it to participants p_0,..., p_t;
6 **for** *each* $p_i \in \{p_0, ..., p_t\}$ **do**
7 p_i samples $y_i \in_R Z_{\overline{n}}$, then computes $(C_*)^{y_i}$ and re-encrypts it, i.e., $(C_*)^{y_i} \leftarrow (C_*)^{y_i} \cdot \mathbf{Enc}(0)$, and sends to p_0;
8 p_0 computes $D_* = \prod_{i=0}^{t} (C_*)^{y_i}$;
9 p_0,..., p_t jointly decrypt D_* by computing $\mathbf{Dec}(D_*)$ to obtain d_* ;
10 If $d_* = 0$, p_0 sets $e_* = 0$; Else, p_0 sets $e_* = 1$;
11 p_0 outputs e_*;
12 **return** e_*

such that each p_j $(j \in [n])$ obtains its own private key sk_j and public key pk. Let $\mathbf{Enc}(\cdot)$, $\mathbf{Dec}(\cdot)$ and $Z_{\overline{n}}$ denote the encryption algorithm, the decryption algorithm and the plaintext space $[\overline{n}]$ for this cryptosystem, respectively, where $\overline{n} = pq$, and p and q are two large primes. We assume that $nM < \overline{n}$.

B. The OR Perturbation Encryption (ORPE)

Before we introduce the remaining three stages, let's introduce our ORPE algorithm (cf. Algorithm 1). This algorithm can be used to enhance participants' privacy. The algorithm can securely compute $\vee_{j\in[n]} b_{j,*}$, where $b_{j,*}$ is a private bit of participant p_j and $*$ indicates the symbols after the subscript letter j. Next, we explain the algorithm. Each participant p_j $(j \in [n])$ has a bit $b_{j,*}$, then it encodes $b_{j,*}$ and obtains $\overline{b}_{j,*}$ (Lines 1-3 of Algorithm 1). Every p_j sends $\overline{b}_{j,*}$ to p_0. Then, p_0 computes C_* by aggregating all the received ciphertexts and its own ciphertext $\overline{b}_{0,*}$ and sends the calculated ciphertext to participants p_0,...,p_t (Line 5 of Algorithm 1). Then each p_i $(i \in [t+1])$ performs the randomization operations (Lines 6-7 of Algorithm 1). After completing the above process, p_0 computes D_* and all p_i $(i \in [t+1])$ jointly decrypt it by using its own sk_i, and p_0 obtains d_* (Lines 8-9 of Algorithm 1). Finally, according to d_*, p_0 computes e_* which is a perturbed data (Lines 10-11 of Algorithm 1).

Algorithm 2: Approximating Algorithm

Input: Private sets $\{S_j\}_{j\in[n]}$, private keys $\{sk_j\}_{j\in[n]}$, public key pk, functions $\{\overline{h}_{\ell'}(\cdot)\}_{\ell'\in[1,m_0]}$
Output: The cardinality estimate $\widetilde{N}$ for the set union $\mathscr{U}$

1 **for** $\ell' = 1$ to m_0 **do**
2 **for** *each* p_j $(j \in [n])$ **do**
3 Participant p_j computes the FM sketch $FM_{j,\ell'} = (b_{j,0,\ell'}, b_{j,1,\ell'}, ..., b_{j,l-1,\ell'})$ based on its own S_j and $\overline{h}_{\ell'}(\cdot)$.
4 **for** $\ell = 0$ to $l-1$ **do**
5 Invoke Algorithm 1: ORPE$(\{p_j, b_{j,\ell,\ell'}, sk_j, pk\}_{j\in[n]})$.
6 Let $e_{\ell,\ell'}$ denote the output of Algorithm 1 obtained by p_0.
7 p_0 computes $z^{\langle \ell' \rangle} = \min\{\ell \mid e_{\ell,\ell'} = 0, \ell \in [l]\}$;
8 p_0 computes $Z = z^{\langle 1 \rangle} + \ldots + z^{\langle m_0 \rangle}$ and $\widetilde{N} = \frac{2^{\frac{Z}{m_0}} - 2^{-\kappa \cdot \frac{Z}{m_0}}}{\phi}$;
9 p_0 announces $\widetilde{N}$;
10 **return** $\widetilde{N}$

C. The Approximating Stage

In this stage, we do not solve the MPSU problem immediately. On the contrary, we first study how to compute an approximate set union cardinality $\widetilde{N}$. Because the approximate set union cardinality $\widetilde{N}$ could contribute to reducing communication and computation complexities. Thus, we first propose an efficient algorithm which computes an approximate set union cardinality $\widetilde{N}$ by using the ORPE technique and FM sketches, and it is shown in Algorithm 2. The input of the algorithm is $\{S_j, sk_j\}_{j\in[n]}$, pk and m_0 functions. The output of the algorithm is the approximate set union cardinality $\widetilde{N}$.

Each participant p_j $(j \in [n])$ creates its own FM sketches by using $\{\overline{h}_{\ell'}(\cdot)\}_{\ell'\in[1,m_0]}$ and S_j (Lines 2-3 of Algorithm 2). Then, all participants invoke Algorithm 1, and p_0 obtains the corresponding output (Lines 5-6 of Algorithm 2). In order to improve accuracy, this algorithm uses m_0 FM sketches (Lines 1-7 of Algorithm 2). After executing m_0 times FM-sketches, p_0 computes the estimator $\widetilde{N}$ for the set union cardinality and sends it to other participants (Lines 8-9 of Algorithm 2).

D. The Global Bloom Filter (GBF) Constructing Stage

In this stage, we will introduce the GBF constructing algorithm, which can construct a global Bloom Filter *GBF* and is presented in Algorithm 3. Based on the GBF, we can compute all elements of the union of all participants' sets. The input of the algorithm for each p_j is the private data set S_j, private key sk_j and public key pk, $\widetilde{N}$ and ε'. The output of the algorithm is the *GBF* .

Note that, in the above stage, we can obtain $\widetilde{N}$ which meets $\Pr\left[|\widetilde{N} - N| \le \varepsilon N\right] \ge 1 - \delta_1$. In order to solve MPSU, we only require that ε is set to $1/2$. So all participants obtain a cardinality estimate $\widetilde{N}$ $(N/2 \le \widetilde{N} \le 3/2N)$. Thus $3N \ge 2\widetilde{N} \ge N$ with probability $1 - \delta_1$. For the Bloom Filter parameters, each p_j sets $k_1 = \log_2(1/\varepsilon')$ and $m = 2\widetilde{N}\log_2 e \cdot \log_2(1/\varepsilon')$, where ε' is a false positive probability (Lines 1-2 of Algorithm 3).

Algorithm 3: GBF Constructing Algorithm

Input: Private sets $\{S_j\}_{j\in[n]}$, private keys $\{sk_j\}_{j\in[n]}$, public key pk, the cardinality estimate $\widetilde{N}$, a false positive probability ε'

Output: Global Bloom Filter *GBF*

1 **for** *each* p_j $(j\in[n])$ **do**
2 $\quad$ p_j computes $k_1=\log_2(1/\varepsilon')$ and $m=2\widetilde{N}\log_2 e\cdot\log_2(1/\varepsilon')$.
3 **for** *each* p_j $(j\in[n])$ **do**
4 $\quad$ p_j computes a local Bloom Filter $BF_j=(b_{j,1},...,b_{j,m})$ using $H=\{h_0(\cdot),...,h_{k_1-1}(\cdot)\}$, as follows;
5 $\quad$ **for** $\ell=1$ to m **do**
6 $\quad\quad$ p_j sets $b_{j,\ell}=0$ (i.e.,local BF initialization).
7 $\quad$ **for** *each* $x\in S_j$ **do**
8 $\quad\quad$ **for** $\ell'=0$ to k_1-1 **do**
9 $\quad\quad\quad$ p_j calculates $h_{\ell'}(x)$ and sets $b_{j,h_{\ell'}(x)}=1$.
10 **for** $\ell=1$ to m **do**
11 $\quad$ Invoke Algorithm 1: ORPE($\{p_j,b_{j,\ell},sk_j,pk\}_{j\in[n]}$).
12 $\quad$ Let e_ℓ denote output of Algorithm 1 obtained by p_0.
13 $\quad$ p_0 sets $\omega_\ell=e_\ell$;
14 p_0 sets $GBF=(\omega_1,...,\omega_m)$ and announces *GBF*.
15 **return** *GBF*

Then, each p_j constructs a local Bloom Filter BF_j by using S_j and H (Lines 3-9 of Algorithm 3). Then, each p_j invokes Algorithm 1 for each bit $b_{j,\ell}$ of BF_j $(\ell\in[1,m])$ and p_0 obtains the corresponding output (Lines 11-13 of Algorithm 3). Then, according to $\{e_\ell \mid \ell\in[1,m]\}$, p_0 obtains the global Bloom Filter $GBF=(\omega_1,...,\omega_m)$ (Line 11 of Algorithm 3). Finally, p_0 sends the *GBF* to other participants (Line 14 of Algorithm 3).

E. The Element Computing Stage

Note that, given $\widetilde{N}$, the global Bloom Filter *GBF* which we construct in the above stage is equivalent to another Bloom Filter which is directly constructed by using the set union, with probability at least $1-2m/\overline{n}$ (cf. the analysis in Section VII). Note that if $\omega_u=0$, for any $u\in[1,m]$, it possibly implies that there do not exist elements mapped to u (with probability $1-2/\overline{n}$) and there must be at least one element otherwise (Detailed analysis can be found in Section VII). The idea of our HE-MPSU protocol is based on a key observation that for Bloom Filter BF, each element in the set union has at least one bit ω_u $(u\in[1,m])$ where only this element is mapped to with probability at least $1-\delta_3$ (δ_3 is a constant between 0 and 1, cf. the subsequent Lemma 2). So we propose an efficient element computing algorithm which can generate the corresponding element which is mapped to for each bit ω_u $(u\in[1,m])$ of the *GBF* and is shown in Fig 3. The input of the algorithm are $GBF,\{p_j,S_j,sk_j,pk\}_{j\in[n]}$, hash functions H and an empty set $\mathscr{U}_1$. The output of the algorithm is the set $\mathscr{U}_1$ which is equal to the set union $\mathscr{U}$, with probability at least $1-\delta$ (see Theorem 2).

Specifically, for each bit ω_u of the GBF, if $\omega_u=0$, this means that there do not exist elements which are mapped to the u-th bit of GBF with probability $1-2/\overline{n}$, and all participants do nothing.

If $\omega_u=1$, each p_j $(j\in[n])$ firstly needs to find the elements $\{x_{j,r_s}\}_{s=1}^{\iota}$ where $h_\ell(x_{j,r_s})=u$ and $\ell\in[k_1]$, and performs the following operations.

1) If there do not exist elements $x_{j,r_s}\in S_j$ such that $h_\ell(x_{j,r_s})=u$ and $\ell\in[k_1]$, i.e., $\iota=0$, p_j calculates two ciphertexts $E_{j,u}=\mathbf{Enc}(0)$ and $N_{j,u}=\mathbf{Enc}(0)$.
2) If there only exists an element $x_{j,r_s}\in S_j$ such that $h_\ell(x_{j,r_s})=u$ and $\ell\in[k_1]$, that is, $\iota=1$, p_j calculates two ciphertexts $E_{j,u}=\mathbf{Enc}(x_{j,r_1})$ and $N_{j,u}=\mathbf{Enc}(1)$.
3) If there exist at least two elements $x_{j,r_s}\in S_i$ such that $h_\ell(x_{j,r_s})=1$, that is, $\iota\geq 2$, p_j calculates two ciphertexts $E_{j,u}=\mathbf{Enc}(e_{j,u})$ and $N_{j,u}=\mathbf{Enc}(1)$, where $e_{j,u}\in_R Z_{\overline{n}}$.

Then, p_j sends $\{E_{j,u},N_{j,u}\}$ to p_0. p_0 computes $E_u=\prod_{j\in[n]}E_{j,u}$ and $N_u=\prod_{j\in[n]}N_{j,u}$ and sends them to the rest of the participants. Each p_j $(j\in[n])$ executes the below operations.

1) If $\iota=0$, p_j calculates the ciphertext $\hat{E}_{j,u}=\mathbf{Enc}(0)$.
2) If $\iota=1$, p_j calculates the ciphertexts $\hat{N}_{j,u}=N_u^{x_{j,r_1}}$ and $\hat{E}_{j,u}=(E_u\hat{N}_{j,u}^{-1})^{r_{j,u}}$, and re-encrypts $\hat{E}_{j,u}$, i.e., $\hat{N}_{j,u}\leftarrow\hat{N}_{j,u}\cdot\mathbf{Enc}(0)$, where $r_{j,u}\in Z_{\overline{n}}$.
3) If $\iota>1$ p_j computes the ciphertext $\hat{E}_{j,u}=E_{j,u}^{r_{j,u}}$ and re-encrypts it, i.e., $\hat{N}_{j,u}\leftarrow\hat{N}_{j,u}\cdot\mathbf{Enc}(0)$, where $r_{j,u}\in Z_{\overline{n}}$.

Then, p_j sends $\hat{E}_{j,u}$ to p_0, and p_0 computes $C_u=\prod_{j\in[n]}\hat{E}_{j,u}$ and sends to participants $\{p_i \mid i\in[t+1]\}$. Participants p_0,..., p_t execute the perturbation operations of Algorithm 1 (Lines 6-11) and p_0 obtains e_u and shares it with other participants.

According to the calculated data e_u, participants consider the following two situations.

Case (1): If $e_u=1$, this means that there exist at least two distinct elements which are mapped to the u-th bit of the GBF (Detailed analysis can be found in Section VII). All participants do nothing.

Case (2): If $e_u=0$, this means that there exists only one element which are mapped to the u-th bit of the GBF with probability at least $1-2/\overline{n}$ (Detailed analysis can be found in Section VII). Each p_j $(j\in[n])$ performs the following operations.

1) If $\iota=0$, p_j computes two ciphertexts $\{\widetilde{E}_{j,u}=\mathbf{Enc}(0),\ \widetilde{N}_{j,u}=\mathbf{Enc}(0)\}$.
2) If $\iota=1$, p_j computes two ciphertexts $\{\widetilde{E}_{j,u}=\mathbf{Enc}(\widetilde{r}_{j,u}x_{j,r_1}),\ \widetilde{N}_{j,u}=\mathbf{Enc}(\widetilde{r}_{j,u})\}$, where $\widetilde{r}_{j,u}\in_R Z_{\overline{n}}$.
3) If $\iota>1$, p_j computes two ciphertexts $\{\widetilde{E}_{j,u}=\mathbf{Enc}(0),\ \widetilde{N}_{j,u}=\mathbf{Enc}(0)\}$.
4) p_j sends $\{\widetilde{E}_{j,u},\widetilde{N}_{j,u}\}$ to p_0.

Then, p_0 calculates two ciphertexts $\widetilde{N}_u=\prod_{j\in[n]}\widetilde{N}_{j,u}$ and $\widetilde{E}_u=\prod_{j\in[n]}\widetilde{E}_{j,u}$, and sends them to participants $\{p_j \mid j\in[t+1]\}$. Each p_j $(j\in[t+1])$ calculates two ciphertexts $\overline{N}_{j,u}=\widetilde{N}_u^{\overline{r}_{j,u}}$ and $\overline{E}_{j,u}=\widetilde{E}_u^{\overline{r}_{j,u}}$, and re-encrypts them, i.e., $\overline{N}_{j,u}\leftarrow\overline{N}_{j,u}\cdot\mathbf{Enc}(0)$ and $\overline{E}_{j,u}\leftarrow\overline{E}_{j,u}\cdot\mathbf{Enc}(0)$, where $\overline{r}_{j,u}\in Z_{\overline{n}}$, and sends them to p_0. Then, p_0 computes $\overline{E}_u=\prod_{j\in[t+1]}\overline{E}_{j,u}$ and $\overline{N}_u=\prod_{j\in[t+1]}\overline{N}_{j,u}$, and sends them to participants $p_0,...,p_t$. All p_i $(i\in[t+1])$ jointly decrypt them to obtain $\overline{e_u}$ and $\overline{n_u}$. Then,

Algorithm 4: Our HE-MPSU Protocol

Input: Private sets $\{S_j\}_{j\in[n]}$, functions $\{\overline{h}_{\ell'}(\cdot)\}_{\ell'\in[1,m_0]}$, a false positive probability ε', a secure parameter λ;
Output: Set $\mathscr{U}_1$

All participants invoke the (n,t)-threshold Paillier cryptosystem based on λ to obtain their own corresponding public and private keys;
All participants run Approximating Algorithm;
All participants run GBF Constructing Algorithm;
All participants run Element Computing Algorithm;
return $\mathscr{U}_1$

p_0 computes $t_u = \overline{e_u}(\overline{n_u})^{-1} \mod \overline{n}$ and adds it to the set $\mathscr{U}_1$, i.e, $\mathscr{U}_1 \leftarrow \mathscr{U}_1 \cup \{t_u\}$.

Finally, p_0 sends the set $\mathscr{U}_1$ to other participants.

For completeness, we formally show our HE-MPSU protocol in Algorithm 4.

VI. OT-PSU

Although our HE-MPSU protocol can be used to solve PSU, it has the same complexities as the previous work in [1]. A natural question is if we can solve this problem faster. Thus, for the two-party case, we further optimize and design a more efficient protocol using OT protocol, i.e., OT-PSU. It avoids using computationally expensive public key operations.

In this section, we introduce our OT-PSU which is formally presented in Fig 4. The protocol is simple and efficient. The basic idea for our OT-PSU is based on the fact $S = S_0 \cup S_1 = S_0 \cup (S \setminus S_0) = S_0 \cup (S_1 \setminus S_0)$. In other words, p_0 only needs to obtain the difference set between the set S_1 of p_1 and its own set S_0. We combine the basic idea and our key observation which is used in our HE-MPSU to devise our OT-PSU protocol.

A. Detailed Protocol Design

Our OT-PSU protocol contains four steps as follows.

Since $k \le N \le 2k$, we do not require the approximating stage which is used in our HE-MPSU. Thus, in the first step, p_0 and p_1 create their own Bloom Filter BF_0 and BF_1, respectively.

In the second step, p_1 will handle each bit of BF_1 as follows. For each $\ell = 1$ to m, p_1 finds the elements $\{x_{1,r_s}\}_{s=1}^{\iota} \in S_1$ where $h_l(x_{1,r_s}) = \ell$ and $l \in [k_1]$, and performs the following operations:

1) If $b_{1,\ell} = 0$, p_1 computes $s_{0,\ell} = s_{1,\ell} = 0$.
2) If $b_{1,\ell} = 1$ and there exists a unique element $\{x_{1,r_s}\}_{s=1}^{\iota} \in S_1$, i.e., $\iota = 1$, such that $h_l(x_{1,r_1}) = \ell$, p_1 sets $s_{0,\ell} = 0$ and $s_{1,\ell} = x_{1,r_1}$.
3) If $b_{1,\ell} = 1$ and there exist at least two elements $\{x_{1,r_s}\}_{s=1}^{\iota} \in S_1$, i.e., $\iota >= 2$, such that $h_l(x_{1,r_s}) = \ell$, p_1 sets $s_{0,\ell} = s_{1,\ell} = 0$.

In the third step, p_0 and p_1 run m OT protocols. Specifically, for each $\ell \in [1,m]$, p_1 acts as the sender with the input $(s_{0,\ell}, s_{1,\ell})$ and p_0 acts as the receiver with the input $1 - b_{0,\ell}$ in the ℓ-th OT. Let x_ℓ denote the corresponding output of p_0 in the ℓ-th OT. After performing the OT protocols, p_0 obtains the data $\{x_1, ..., x_m\}$.

In the last step, p_0 computes $\mathscr{U}_1 = S_0 \cup \{x_1, ..., x_m\} \setminus \{0\}$ (note that $S_0, S_1 \subseteq [1, M]$) and announces $\mathscr{U}_1$. With probability $1 - \delta_3$ ($N, m \in O(k)$), we have $\mathscr{U}_1 = S_0 \cup S_1$ (Detailed analysis can be found in Section VII).

VII. Correctness, Security and Complexity Analysis

In this section, we will analyze the correctness, security and complexity of our two protocols in detail, respectively.

A. Correctness

We first consider the HE-MPSU protocol. For the approximating algorithm about the set union cardinality N, we only need to consider whether or not our algorithm can accurately get the FM sketches of the set union. Because the correctness of the estimated value for N can be easily obtained from Lemma 1. We have a theorem as follows.

Theorem 1. *In the approximating stage, participants can obtain the m_0 FM sketches of the set union $\mathscr{U}$, with probability at least $1 - \delta_2$, where $\delta_2 = 2lm_0/\overline{n}$.*

Proof. W.l.o.g., we suppose that the FM sketch $FM = (g_0, ..., g_{l-1})$ is produced by using the set union $\mathscr{U}$ and $\overline{h}(\cdot)$ directly. We mainly consider two different cases (1) $g_s = 0$ and (2) $g_s = 1$, for a fixed $s \in [l]$.

Case (1): When $g_s = 0$, we know that there are no elements x so that $\overline{h}(x) = s$ for any $x \in \mathscr{U}$. Each p_i ($i \in [n]$) computes FM sketches $FM_i = (b_{i,0}, ..., b_{i,s}, ..., b_{i,l-1})$. Then, each p_i computes $\overline{b}_{i,s} = \mathbf{Enc}(0)$ (since $b_{i,s} = 0$) and sends it to p_0. Then, p_0 obtains $C_s = \prod_{j\in[n]} \overline{b}_{j,s} = \prod_{j\in[n]} \mathbf{Enc}(0) = \mathbf{Enc}(0)$ based on homomorphic properties. Next, after $p_0, ..., p_t$ perform the perturbation operations in lines 6-7 of Algorithm 1, p_0 can obtain $D_s = \prod_{j\in[t+1]} C_s^{y_j} = \prod_{j\in[t+1]} \mathbf{Enc}(0) = \mathbf{Enc}(0)$ (Line 8 of Algorithm 1). p_0 can obtain $d_s = \mathbf{Dec}(D_s) = 0$ by using the decryption algorithm. Thus we have $e_s = 0$ and must obtain a correct result, in this case.

Case (2): When $g_s = 1$, we know that there is at least one element x so that $\overline{h}(x) = s$ for $x \in \mathscr{U}$. W.l.o.g., we suppose that participants p_0 and p_1 have such an element. Then participant p_0 (p_1) samples $e_{0,s} \in_R Z_{\overline{n}}$ ($e_{1,s} \in_R Z_{\overline{n}}$) and computes $\overline{b}_{0,s} = \mathbf{Enc}(e_{0,s})$ ($\overline{b}_{1,s} = \mathbf{Enc}(e_{1,s})$) and sends it to p_0. Similar to the analysis in **Case (1)**, p_0 can obtain $d_s = (e_{0,s} + e_{1,s}) \sum_{i=0}^{t} y_i$. Because $y_i \in_R Z_{\overline{n}}$ for any $i \in \{0, ..., t\}$, we have $d_s = 0$ if and only if $e_{0,s} + e_{1,s} \mod \overline{n} = 0$ or $\sum_{i=0}^{t} y_i \mod \overline{n} = 0$, with probability at most $2/\overline{n}$. Thus, in this case, we obtain a correct result, with probability at least $1 - 2/\overline{n}$.

By a union bound, we can obtain an FM sketch of the set union $S_0 \cup S_1 \cup \cdots \cup S_{n-1}$, with probability at least $1 - 2l/\overline{n}$. Thus, we can obtain the m_0 FM sketches of the set union $\mathscr{U}$, with probability at least $1 - 2lm_0/\overline{n}$.

□

Next, we consider the GBF and have the below lemma.

Lemma 2. *For the global Bloom Filter, all elements in $\mathscr{U}$ have at least one bit where the same element is mapped to with*

Element Computing Algorithm

Input: Global Bloom Filter $GBF = (\omega_1, ..., \omega_m)$, $\{p_j, S_j, sk_j, pk\}_{j\in[n]}$, hash functions $H = \{h_i(\cdot)\}_{i\in[k_1]}$ and an empty set $\mathscr{U}_1$

Output: Set union $\mathscr{U}_1$.

1) For each $u = 1$ to m, do:
   a) If $\omega_u = 0$, all participants do nothing.
   b) If $\omega_u = 1$, participants perform the following operations.
      i) Each p_j $(j \in [n])$ finds the elements $\{x_{j,r_s}\}_{s=1}^{\iota} \in S_j$ where $h_\ell(x_{j,r_s}) = u$ and $\ell \in [k_1]$, and computes the following ciphertexts:
         A) If $\iota = 0$, p_j computes two ciphertexts $\{E_{j,u} = \mathbf{Enc}(0), N_{j,u} = \mathbf{Enc}(0)\}$.
         B) If $\iota = 1$, p_j computes two ciphertexts $\{E_{j,u} = \mathbf{Enc}(x_{j,r_1}), N_{j,u} = \mathbf{Enc}(1)\}$.
         C) If $\iota > 1$, p_j computes two ciphertexts $\{E_{j,u} = \mathbf{Enc}(e_{j,u}), N_{j,u} = \mathbf{Enc}(1)\}$, where $e_{j,u} \in_R Z_{\overline{n}}$.
         D) p_j sends $\{E_{j,u}, N_{j,u}\}$ to p_0.
      ii) p_0 computes $\{E_u = \prod_{j\in[n]} E_{j,u}, N_u = \prod_{j\in[n]} N_{j,u}\}$ and sends to the rest of the participants.
      iii) Every p_j $(j \in [n])$ computes the following ciphertexts:
         A) If $\iota = 0$, p_j calculates the ciphertext $\hat{E}_{j,u} = \mathbf{Enc}(0)$.
         B) If $\iota = 1$, p_j calculates the ciphertexts $\hat{N}_{j,u} = N_u^{x_{j,r_1}}$ and $\hat{E}_{j,u} = (E_u \hat{N}_{j,u}^{-1})^{r_{j,u}}$, and re-encrypts $\hat{E}_{j,u}$, i.e., $\hat{E}_{j,u} \leftarrow \hat{E}_{j,u} \cdot \mathbf{Enc}(0)$, where $r_{j,u} \in Z_{\overline{n}}$.
         C) If $\iota > 1$ p_j computes the ciphertext $\hat{E}_{j,u} = E_{j,u}^{r_{j,u}}$, and re-encrypts $\hat{E}_{j,u}$ i.e., $\hat{E}_{j,u} \leftarrow \hat{E}_{j,u} \cdot \mathbf{Enc}(0)$, where $r_{j,u} \in Z_{\overline{n}}$.
         D) p_j sends $\hat{E}_{j,u}$ to p_0.
      iv) p_0 computes $C_u = \prod_{j\in[n]} \hat{E}_{j,u}$ and sends to participants $p_0, ..., p_t$.
      v) p_0,..., p_t execute the perturbation operations of Algorithm 1 (Lines 6-11) to obtain data e_u, and p_0 sends it to other participants.
      vi) If $e_u = 1$, all participants do nothing.
      vii) If $e_u = 0$, participants perform the following operations.
         A) Each p_j $(j \in [n])$ computes the following ciphertexts:
            (1) If $\iota = 0$, p_j computes two ciphertexts $\{\widetilde{E}_{j,u} = \mathbf{Enc}(0), \widetilde{N}_{j,u} = \mathbf{Enc}(0)\}$.
            (2) If $\iota = 1$, p_j computes two ciphertexts $\{\widetilde{E}_{j,u} = \mathbf{Enc}(\widetilde{r}_{j,u} x_{j,r_1}), \widetilde{N}_{j,u} = \mathbf{Enc}(\widetilde{r}_{j,u})\}$, where $\widetilde{r}_{j,u} \in_R Z_{\overline{n}}$.
            (3) If $\iota > 1$, p_j computes two ciphertexts $\{\widetilde{E}_{j,u} = \mathbf{Enc}(0), \widetilde{N}_{j,u} = \mathbf{Enc}(0)\}$.
            (4) p_j sends $\{\widetilde{E}_{j,u}, \widetilde{N}_{j,u}\}$ to p_0.
         B) p_0 calculates two ciphertexts $\widetilde{N}_u = \prod_{j\in[n]} \widetilde{N}_{j,u}$ and $\widetilde{E}_u = \prod_{j\in[n]} \widetilde{E}_{j,u}$, and sends them to participants $\{p_j \mid j \in [t+1]\}$.
         C) Each p_j $(j \in [t+1])$ calculates two ciphertexts $\overline{N}_{j,u} = \widetilde{N}_u^{\overline{r}_{j,u}}$ and $\overline{E}_{j,u} = \widetilde{E}_u^{\overline{r}_{j,u}}$, and re-encrypts them, i.e., $\overline{N}_{j,u} \leftarrow \overline{N}_{j,u} \cdot \mathbf{Enc}(0)$ and $\overline{E}_{j,u} \leftarrow \overline{E}_{j,u} \cdot \mathbf{Enc}(0)$, and sends to p_0, where $\overline{r}_{j,u} \in Z_{\overline{n}}$.
         D) p_0 computes $\overline{E}_u = \prod_{j\in[t+1]} \overline{E}_{j,u}$ and $\overline{N}_u = \prod_{j\in[t+1]} \overline{N}_{j,u}$ and sends to participants $p_0, ..., p_t$.
         E) All p_i $(i \in [t+1])$ jointly decrypt $\overline{E}_u$ and $\overline{N}_u$ to obtain $\overline{e_u}$ and $\overline{n_u}$, respectively.
         F) p_0 computes $t_u = \overline{e_u}(\overline{n_u})^{-1} \mod \overline{n}$ and adds it to the set $\mathscr{U}_1$, i.e., $\mathscr{U}_1 \leftarrow \mathscr{U}_1 \cup \{t_u\}$.
2) p_0 announces $\mathscr{U}_1$.

Fig. 3. The descriptions of the element computing algorithm.

probability at least $1 - \delta_3$, *where* $p_2 = 1 - (1 - 1/m)^{k_1(N-1)}$ *and* $\delta_3 = p_2^{k_1} \times \left(1 + O(\frac{k_1}{p_2}\sqrt{\frac{\ln m - k_1 \ln p_2}{m}})\right)$ *is negligible in* k_1.

The proof of this lemma is the same as that of Theorem 1 in [29] and we omit it here. What we want to emphasize is that the use of Bloom Filter in their protocols is similar to that of the previous works [24], [1], [18], [5] i.e., their protocols need to use all bits which are mapped to by using k_1 hash functions for each element.

Theorem 2. *In our HE-MPSU protocol, given* $nM < \overline{n}$, *all participants can obtain the set union* $\mathscr{U}$ *(i.e.,* $\mathscr{U}_1 = \mathscr{U}$*), with probability at least* $1 - \delta$, *where* $\delta \leq \delta_1 + \delta_2 + \delta_3 + \delta_4$ *and* $\delta_4 = 4m/\overline{n} + 2m(1/q + 1/p)$.

Theorem 3. *In our OT-PSU protocol, participants* p_0 *and* p_0 *can obtain set union* $S = S_0 \cup S_1$, *with probability at least* $1 - \delta$, *where* $\delta = \delta_3$ *and* $N, m \in O(k)$.

Due to space constraints, detailed proof of the above two theorems can be found in our technical report [39].

B. Security

Theorem 4. *Our HE-MPSU protocol is secure if the* (n,t)*-threshold Paillier cryptosystem is semantically secure.*

Theorem 5. *Our OT-PSU protocol is secure if the OT protocols are secure.*

The OT-PSU Protocol

Input: Participants p_0 and p_1 which hold private data sets S_0 and S_1, respectively. Hash functions $H = \{h_0(\cdot), ..., h_{k_1-1}(\cdot)\}$.

Output: Participants p_0 and p_1 learn the set union $S_0 \cup S_1$.

1) For all $j \in [2]$, p_j computes a Bloom Filter $BF_j = (b_{j,1}, ..., b_{j,m})$, as shown below:
   a) For all $\ell \in [1,m]$, p_j sets $b_{j,\ell} = 0$ (i.e., BF initialization).
   b) For all $x \in S_j$ and all $l \in [k_1]$, p_j calculates $h_l(x)$ and sets $b_{j,h_l(x)} = 1$.
2) p_1 computes m messages pair $\{s_{0,\ell}, s_{1,\ell}\}_{\ell \in [1,m]}$ as shown below:
   a) For each $\ell = 1$ to m, p_1 finds the elements $\{x_{1,r_s}\}_{s=1}^{\iota} \in S_1$, where $h_l(x_{1,r_s}) = \ell$ and $l \in [k_1]$, and performs the following operations:
      i) If $b_{1,\ell} = 0$, p_1 sets $s_{0,\ell} = s_{1,\ell} = 0$.
      ii) If $b_{1,\ell} = 1$ and $\iota = 1$, p_1 sets $s_{0,\ell} = 0$ and $s_{1,\ell} = x_{1,r_1}$.
      iii) If $b_{1,\ell} = 1$ and $\iota >= 2$, p_1 sets $s_{0,\ell} = s_{1,\ell} = 0$.
3) For each $\ell \in [1,m]$, p_0 and p_1 run an OT protocol as follows:
   a) p_1 acts as the sender with the input $(s_{0,\ell}, s_{1,\ell})$ for the OT.
   b) p_0 acts as the receiver with the input $1 - b_{0,\ell}$ for the OT.
   Let x_ℓ denote the corresponding output of the OT obtained by p_0.
4) p_0 computes $\mathscr{U}_1 = S_0 \cup \{x_1, ..., x_m\} \setminus \{0\}$ and announces $\mathscr{U}_1$.

Fig. 4. The descriptions of the OT-PSU protocol.

Due to space constraints, detailed proof of the above two theorems can be found in our technical report [39].

Remark. The security requirement of our protocols is that no participants can learn any extra knowledge other than what can be obtained from the set union. Let's look at a simple example. Suppose that there are two participants p_0 and p_1, which have $S_0 = \{x_1, x_2, x_3\}$ and $S_1 = \{x_2, x_4\}$, respectively. After executing our OT-PSU protocol, p_0 and p_1 obtain the set union $\mathscr{U} = \{x_1, x_2, x_3, x_4\}$. Obviously, by using sets S_0 and $\mathscr{U}$, p_0 learns the knowledge that p_1 has x_4. According to Definition 2, this knowledge is allowed to be learned, since it is inferred from the set union. But other than that, p_0 cannot learn any extra knowledge from our protocol, such as whether or not p_1 holds elements x_1, x_2 and x_3.

C. Complexities

1) Communication Complexity: For the HE-MPSU protocol, in the system setup stage, the communication complexity is $O(n\lambda)$. Next, in the approximating stage, each p_i needs to send m_0 l-bit FM sketches. And for each bit, each p_i produces ciphertexts of length $O(\lambda)$. Thus, the communication complexity is $O(m_0 nl\lambda)$ in the stage. Similarly, the communication complexity of the remaining stages is $O(nm\lambda)$. In summary, for the given constant δ, the total communication complexity is $O(nN\lambda)$ since m_0 is a constant, $l = \lceil \log M \rceil$ and $m = O(N)$.

For our OT-PSU protocol, the communication complexity is $O(k\lambda)$ since the communication complexity of one OT protocol is $O(\lambda)$ and we need to run $O(k)$ OT protocols, for given constant δ.

2) Round Complexity: For the HE-MPSU protocol, our system setup stage can be done in $O(1)$ rounds. The approximating stage requires $O(1)$ rounds since performing Algorithm 1 requires $O(1)$ rounds and for each step, all messages required to send by participants can be transmitted in parallel. Similarly, for the rest of the stages, they need $O(1)$ rounds. Thus, the total round complexity is $O(1)$.

For our OT-PSU protocol, it is not hard to see that round complexity is $O(1)$ since OT protocols can be done in constant rounds and, for each step, all messages required to send by participants can be transmitted in parallel.

3) Computation Complexity: For the HE-MPSU protocol, we mainly consider modular exponentiation operations. When each participant generates one ciphertext by using $\mathbf{Enc}(\cdot)$, it needs $O(1)$ modular exponentiation operations. For each $j \in [t+1]$, p_j needs $O(1)$ modular exponentiation operations to decrypt one ciphertext. According to the above analysis, we have that each participant generates and decrypts at most $O(N)$ ciphertexts. Thus, the number of modular exponentiations operations is $O(nN)$.

For the OT-PSU protocol, the number of modular exponentiations operations is $O(\lambda)$ thanks to OT extensions. The number of exponentiations in this protocol is independent of the size of the data sets.

VIII. Discussion on Different Set Sizes of Each Participant

In this section, we discuss the case where the size of each participant's data set is different.

We mainly consider the multi-party case, i.e., our HE-MPSU protocol. It is not difficult to see that the operation of each participant in the system setup stage is independent of the size of the participants' sets. In the approximating stage, each participant generates and sends m_0 FM-Sketches, and each FM-Sketch contains $l = \lceil \log M \rceil$ ciphertexts. Note that M (or the universal set $[1,M]$) is public. The number of messages sent by each participant in this stage is independent of the size of its own set. In the GBF constructing stage, participants construct a GBF, and the GBF contains $O(N)$ ciphertexts. Note that N is public since it is easily calculated from $|\mathscr{U}|$. The number of messages sent by each participant in this stage is also independent of the size of its own set. Similarly, in the

element computing stage, the number of messages sent by each participant is also independent of the size of its own set since each participant sends $O(N)$ ciphertexts. Therefore, our HE-MPSU protocol can be directly applied to the case where the sets of participants are of different sizes, and no additional operations need to be added. In other words, our protocol naturally hides the size of each participant's set.

However, in the previous work [12], [16], the number of messages sent by each participant is related to the size of its own set. If the size of each participant's set is privacy information and needs to be protected, their schemes usually require an additional step: padding operation. Namely, each participant must add dummy elements to its private data set. Obviously, this operation will increase the communication and computation overheads.

IX. Conclusion

In this paper, we have revisited the classic MPSU problem and proposed nearly optimal protocols, which have the lowest communication, computation and round complexities compared with the existing work. Our protocols can securely obtain the set union with probability at least $1-\delta$ where δ is a small constant between 0 and 1. Our protocols are based on our key observation, i.e., for the Bloom Filter $BF = (b_1, ...b_m)$ produced by a set union, each element in the set union has at least one bit b_i $(i \in [1,m])$ where only this element is mapped to with probability at least $1-\delta_3$ (δ_3 is also a constant between 0 and 1). In our future work, we will investigate the classic MPSU problem in the malicious model.

X. Acknowledgments

The authors thank the anonymous reviewers for their helpful suggestions to improve the presentation of the paper. This work is supported in part by the National Natural Science Foundation of China, under Grant No. 61972447, 61832006.

References

[1] A. Davidson and C. Cid, "An efficient toolkit for computing private set operations," in *Proc. ACISP*. Springer, 2017, pp. 261–278.

[2] L. Kissner and D. Song, "Privacy-preserving set operations," in *Proc. CRYPTO*. Springer, 2005, pp. 241–257.

[3] M. Blanton and E. Aguiar, "Private and oblivious set and multiset operations," *Int. J. Inf. Secur*, vol. 15, no. 5, pp. 493–518, 2016.

[4] J. Brickell and V. Shmatikov, "Privacy-preserving graph algorithms in the semi-honest model," in *Proc. ASIACRYPT*. Springer, 2005, pp. 236–252.

[5] R. Egert, M. Fischlin, D. Gens, S. Jacob, M. Senker, and J. Tillmanns, "Privately computing set-union and set-intersection cardinality via bloom filters," in *Proc. ACISP*. Springer, 2015, pp. 413–430.

[6] A. Lenstra and T. Voss, "Information security risk assessment, aggregation, and mitigation," in *Proc. ACISP*, 2004.

[7] V. Kolesnikov, M. Rosulek, N. Trieu, and X. Wang, "Scalable private set union from symmetric-key techniques," in *Proc. ASIACRYPT*. Springer, 2019, pp. 636–666.

[8] M. Kantarcioglu and C. Clifton, "Privacy-preserving distributed mining of association rules on horizontally partitioned data," *IEEE TKDE*, vol. 16, no. 9, pp. 1026–1037, 2004.

[9] A. C. Yao, "Protocols for secure computations," in *Proc. FOCS*. IEEE, 1982, pp. 160–164.

[10] S. Goldwasser, "How to play any mental game, or a completeness theorem for protocols with an honest majority," *In Proc. STOC*, pp. 218–229, 1987.

[11] M. J. Freedman, K. Nissim, and B. Pinkas, "Efficient private matching and set intersection," in *Proc. EUROCRYPT*. Springer, 2004, pp. 1–19.

[12] K. Frikken, "Privacy-preserving set union," in *Proc. ACNS*. Springer, 2007, pp. 237–252.

[13] Y. Sang and H. Shen, "Efficient and secure protocols for privacy-preserving set operations," *ACM TISSEC*, vol. 13, no. 1, p. 9, 2009.

[14] D. P. Woodruff and Q. Zhang, "When distributed computation is communication expensive," *Distrib. Comput.*, vol. 30, no. 5, pp. 309–323, 2017.

[15] W. Wang, S. Li, J. Dou, and R. Du, "Privacy-preserving mixed set operations," *Inf. Sci.*, vol. 525, pp. 67–81, 2020.

[16] J. H. Seo, J. H. Cheon, and J. Katz, "Constant-round multi-party private set union using reversed laurent series," in *Proc. PKC*. Springer, 2012, pp. 398–412.

[17] G. Garimella, P. Mohassel, M. Rosulek, S. Sadeghian, and J. Singh, "Private set operations from oblivious switching," in *Proc. PKC*. Springer, 2021, pp. 591–617.

[18] D. Many, M. Burkhart, and X. Dimitropoulos, "Fast private set operations with sepia," *ETZ G93*, 2012.

[19] K. Shishido and A. Miyaji, "Efficient and quasi-accurate multiparty private set union," in *Proc. SMARTCOMP*, 2018, pp. 309–314.

[20] C. Dong and G. Loukides, "Approximating private set union/intersection cardinality with logarithmic complexity," *IEEE TIFS*, vol. 12, no. 11, pp. 2792–2806, 2017.

[21] Y. Zhang, Q. Chen, and S. Zhong, "Privacy-preserving data aggregation in mobile phone sensing," *IEEE TIFS*, vol. 11, no. 5, pp. 980–992, 2016.

[22] X. Gong, Q.-S. Hua, L. Qian, D. Yu, and H. Jin, "Communication-efficient and privacy-preserving data aggregation without trusted authority," in *Proc. INFOCOM*. IEEE, 2018, pp. 1250–1258.

[23] C. Hazay and K. Nissim, "Efficient set operations in the presence of malicious adversaries," in *Proc. PKC*. Springer, 2010, pp. 312–331.

[24] A. Davidson and C. Cid, "Computing private set operations with linear complexities." *IACR Cryptology ePrint Archive*, vol. 2016, p. 108, 2016.

[25] G. Oded, "Foundations of cryptography. basic applications, vol. 2," 2004.

[26] P. Flajolet and G. N. Martin, "Probabilistic counting algorithms for data base applications," *J. Comput. Syst. Sci.*, vol. 31, no. 2, pp. 182–209, 1985.

[27] B. Scheuermann and M. Mauve, "Near-optimal compression of probabilistic counting sketches for networking applications," in *Proc. DIALM-POMC*. ACM, 2007.

[28] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Commun. ACM*, vol. 13, no. 7, pp. 422–426, 1970.

[29] C. Dong, L. Chen, and Z. Wen, "When private set intersection meets big data: an efficient and scalable protocol," in *Proc. CCS*, 2013, pp. 789–800.

[30] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Proc. EUROCRYPT*. Springer, 1999, pp. 223–238.

[31] P.-A. Fouque, G. Poupard, and J. Stern, "Sharing decryption in the context of voting or lotteries," in *Proc. FC*. Springer, 2000, pp. 90–104.

[32] C. Hazay and M. Venkitasubramaniam, "Scalable multi-party private set-intersection," in *Proc. PKC*. Springer, 2017, pp. 175–203.

[33] Y. Zhang, Q. Chen, and S. Zhong, "Efficient and privacy-preserving min and k th min computations in mobile sensing systems," *IEEE TDSC*, vol. 14, no. 1, pp. 9–21, 2015.

[34] A. C. Yao, "How to generate and exchange secrets," in *Proc. FOCS*. IEEE, 1986, pp. 162–167.

[35] S. Even, O. Goldreich, and A. Lempel, "A randomized protocol for signing contracts," *Commun. ACM*, vol. 28, no. 6, pp. 637–647, 1985.

[36] Y. Ishai, J. Kilian, K. Nissim, and E. Petrank, "Extending oblivious transfers efficiently," in *Proc. CRYPTO*. Springer, 2003, pp. 145–161.

[37] G. Asharov, Y. Lindell, T. Schneider, and M. Zohner, "More efficient oblivious transfer and extensions for faster secure computation," in *Proc. CCS*, 2013, pp. 535–548.

[38] V. Kolesnikov and R. Kumaresan, "Improved ot extension for transferring short secrets," in *Proc. CRYPTO*. Springer, 2013, pp. 54–70.

[39] X. Gong, Q.-S. Hua, and H. Jin, "Nearly optimal protocols for computing multi-party private set union." Technical report. [Online]. Available: https://qiangshenghua.github.io/papers/iwqos22full.pdf.