Multi-party private set intersection protocols for practical applications

Bay, Asli; Erkin, Zeki; Alishahi, Mina; Vos, Jelle

DOI
10.5220/0010547605150522

Publication date
2021

Document Version
Accepted author manuscript

Published in
Proceedings of the 18th International Conference on Security and Cryptography, SECRYPT 2021

Multi-party Private Set Intersection Protocols for Practical Applications

Asli Bay[1], Zeki Erkin[2,4], Mina Alishahi[3] and Jelle Vos[2]
[1]*Antalya Bilim University, Turkey*
[2]*Delft University of Technology, The Netherlands*
[3]*Eindhoven University of Technology, The Netherlands*
[4]*Radboud University, The Netherlands*
asli.bay@antalya.edu.tr, z.erkin@tudelft.nl, m.sheikhalishahi@tue.nl, j.v.vos@student.tudelft.nl

Keywords: Multi-party private set intersection, bit-set representation, Threshold PKE, privacy-preserving protocols.

Abstract: Multi-party Private Set Intersection (MPSI) is an attractive topic in research since a practical MPSI protocol can be deployed in several real-world scenarios, including but not limited to finding the common list of customers among several companies or privacy-preserving analyses of data from different stakeholders. Several solutions have been proposed in the literature however, the existing solutions still suffer from performance related challenges such as long run-time and high bandwidth demand, particularly when the number of involved parties grows. In this paper, we propose a new approach based on threshold additively homomorphic encryption scheme, *e.g.*, Paillier, which enables us to process the bit-set representation of sets under encryption. By doing so, it is feasible to securely compute the intersection of several data sets in an efficient manner. To prove our claims on performance, we compare the communication complexity of our approach with the existing solutions and show performance test results. We also show how the proposed protocol can be extended to securely compute other set operations on multi-party data sets.

1 Introduction

Multi-party Private Set Intersection (MPSI) is the process of finding the common elements in several data sets without revealing any information about the data but the intersection itself. Formally, t parties $P_1, \ldots, P_t$ owning the sets $S_{P_1}, \ldots, S_{P_t}$, respectively, are interested in finding $S_{P_1} \cap \ldots \cap S_{P_t}$ without revealing their data sets. Multi-party private set intersection has wide application in real-world scenarios: MPSI can be used among several commercial companies to find the intersection of customer lists. The list of common customers can be used to plan promotions for such customers (Cheon et al., 2012); MPSI can be used among the community of medical professionals to find out the patients of a hospital who have participated in the medical tests of different research labs (Cao et al., 2017); MPSI can also be used in multi-party access control, where several co-owners of a common content each specify a set of users who are permitted to access data. The ones in the intersection are allowed to access the content (Sheikhalishahi et al., 2019).

Given the relevance of the problem, several works have been proposed in the literature. While the proposed approaches provide solutions for the MPSI problem in different scenarios, they mainly suffer from computation and communication complexity that grow when the number of involved parties increases. To solve this issue, in this study we propose an efficient MPSI protocol which its complexity is more efficient than the existing ones in terms of the number of parties. The main intuition behind protocol is that the data sets are converted to a *bit-set* representation, *i.e.*, a vector of bits is assigned to each data set S in which the i'th element of this bit-vector (bit-set) is equal to 1, if the i'th element of an ordered *domain* of elements belongs to S, and 0 otherwise. This new representation makes the protocols' computation and communication complexity (mainly) dependent on domain size. Our proposed approach is an effective MPSI solution when the domain size is small enough and the number of parties increases.

The growth of the number of parties is particularly an issue in certain real-world applications. For instance the problem of recommending items from a limited-size catalogue to customers: There is a growing number of customers but a limited size catalogue. An MPSI protocol can identify the items that a group of customers have all bought or looked at. In general, these protocols are particularly interesting for creat-

ing social groups with similar interests.

The design of our MPSI protocol relies on threshold additively homomorphic encryption. Threshold schemes are important in settings in which no individual party knows the secret key, *i.e.*, it provides a strong security guarantee. An additive homomorphic scheme supports the addition of two ciphertexts without needing them to be decrypted, which is an appealing property for MPSI protocol enabling the server to find the common elements in data sets without decrypting the individual messages. Our experimental results show that for a domain of 2^8 elements, where each party owns 16 elements out of domain, the total runtime stays within 16 seconds for a group of $t = 50$ parties (with collusion threshold of $\ell = \lfloor \frac{t}{2} \rfloor$). This result is significantly better than the most efficient MPSI protocol in the literature proposed by Kolesnikov et. al. (Kolesnikov et al., 2017). The contribution of this study can be summarized as follows:

- Based on the bit-set representation of data sets and threshold additively homomorphic encryption, we design an efficient MPSI protocol: MPSI-1.
- We improve the communication complexity in MPSI-2, which reduces the server interaction with clients from $O(dt)$ to $O(d\ell)$, where $\ell < t$.
- We provide the correctness, complexity analysis, and formal simulation-based security proof of both protocols in semi-honest adversary model.
- We show how to extend our solution to other privacy-preserving set operations, *e.g.* set union, threshold set intersection, and set subtraction.
- We provide a theoretical comparison of our protocols' complexity with the existing MPSI protocols in the literature.
- We validate the efficiency of our MPSI protocol implementation by comparing its runtime with the most efficient MPSI approach in the literature.

The rest of this paper is organized as follows: Section 2 discusses related work.Section 3 describes preliminary background. Section 4 presents an extension of Ruan's PSI protocol, and Section 5 improves it based on the bit-set representation. Section 6 provides experimental results of our protocol. Section 7 concludes and provides directions for future work.

2 Related Work

In recent years, many works in literature has been devoted to private set intersection. The first research investigated the use of cryptographic techniques in the context of *two-party* private set intersection (PSI). For instance, in (Dong et al., 2013), a new approach named oblivious Bloom intersection is proposed which uses the combination of symmetric key operations and garbled Bloom filters, or in (Cao et al., 2017), which uses a combination of commutative encryption and hash-based commitments. Although these works provide effective solutions for addressing the PSI problem, they are not applicable for Multi-party PSI. Thus, further research attempts to address PSI in the *multi-party* setting.

Table 1: The comparison of previous designs with ours; where n is the number of elements in a data set; d is the domain size; t is number of parties; ℓ is the *threshold* of Homomorphic PKE; κ and λ are the computational and statistical security parameter; Ω is the size of each bin of Garbled Bloom filter; and $\log|X|$ is the bit-size of ciphertext X.

Protocol	Communication bits		Computation encryptions	
	Server	Client	Server	Client
(Kissner and Song, 2005)	$O(\lambda t^2 n \log\|X\|)$		$O(\ell t n^2)$	
(Cheon et al., 2012)	$O(\lambda t^2 n)$		$O(tn)$	
(Miyaji et al., 2015)	$O(\lambda t n \log\|X\|)$	$O(\lambda t n \log\|X\|)$	$O(\lambda n)$	$O(\lambda n)$
(Hazay et al., 2017)	$O(tn\lambda)$	$O(n\lambda)$	$O(tn \log n)$	$O(n)$
(Kolesnikov et al., 2017)	$O(tn\lambda)$	$O(n\ell\lambda)$	$O(t\kappa)$	$O(\ell\kappa)$
(Inbar et al., 2018)	$O(tn\kappa\Omega)$	$O(tn\kappa\Omega)$	$O(tn\kappa\Omega)$	$O(tn\kappa\Omega)$
MPSI-1 (Section 4)	$O(dt \log\|X\|)$	$O(d \log\|X\|)$	$O(d)$	$O(d)$
MPSI-2 (Section 5)	$O(d\ell \log\|X\|)$	$O(d \log\|X\|)$	$O(d)$	$O(d)$

In (Kissner and Song, 2005), with the use of mathematical properties of polynomials, the composable protocols over multiset operations are designed. In polynomial computations, the multiplication (over encrypted coefficients) generally imposes quadratic computational and communicational complexities in the degree of polynomials. To mitigate this issue, (Cheon et al., 2012) propose an approach in which a polynomial $f(x)$ is represented by several points on the curve $y = f(x)$. To solve the MPSI problem, a scalable and flexible approach is proposed in (Miyaji and Nishida, 2015) in which the data set size of each party is independent to each other and the computational complexity is independent to the number of parties. A new approach for MPSI using two-party set intersection protocols is developed in (Hazay and Venkitasubramaniam, 2017). The proposed approach addresses the challenges of using two-party protocols for intermediate computations without violating the privacy of the multi-party construction. A new paradigm for MPSI using oblivious evaluation of a programmable pseudorandom function (OPPRF) is proposed in (Kolesnikov et al., 2017). The proposed protocols avoid expensive public-key operations and are secure in the presence of any number of semi-honest participants. In (Inbar et al., 2018), two protocols based on Bloom filters and threshold Homomorphic Public Key Encryptions (PKE) are proposed in the multi-party setting. Table 1 reports the complexity of existing MPSI approaches and ours *i.e.*, MPSI-1 and MPSI-2. The MPSI-2 is the optimized version of MPSI-1 in which the communication complexity of server has been reduced from $O(dt \log|X|)$ to $O(d\ell \log|X|)$ where $\ell < t$.

3 Preliminaries

3.1 Security Model

The security of our protocols are proven in the *semi-honest* model with static adversaries. So, the number of corrupted parties are known before the protocol starts and they follow the protocol honestly. Due to space limitations, we briefly mention the security concepts and give a reference link to the reader. The proofs of the protocols are based on the the security definition of the semi-honest security for deterministic functionalities in (Goldreich and Warning, 1998).

In our protocols, we use the threshold variant of Paillier PKE scheme proposed in (Fouque et al., 2000) which is additively homomorphic that is, if given two ciphertexts $c_1 = \mathsf{Enc}(\mathsf{pk}, M_1)$ and $c_2 = \mathsf{Enc}(\mathsf{pk}, M_2)$ one can efficiently compute $\mathsf{Enc}(\mathsf{pk}, M_1 + M_2)$ without the knowledge of the secret key. Also, we use the approach proposed in (Hazay and Venkitasubramaniam, 2017) which provides an extension of the decryption algorithm of an additively homomorphic threshold PKE that allows the involved parties to learn whether a ciphertext is encryption of zero or not, but nothing else. We will call such decryption algorithm "*decryption-to-zero*" and denote it by ShDec0. This property can typically be achieved by randomizing the ciphertext by each of the involved parties, combining the results in a new value, and jointly decrypting this obtained value.

3.2 Bit-set Representation

We adopt the data set representation in terms of bit vectors as employed by (Ruan et al., 2019), namely *bit-set* representation. Formally, assume that parties' data sets are selected from a fixed and ordered domain that consists of d elements $\mathcal{S} = \{s_1, s_2, \ldots, s_d\}$. Let a bit vector $\overrightarrow{B} = (b_1, b_2, \ldots, b_d)$ denotes a set S which is a subset of $\mathcal{S}$, where if $b_j = 1, 1 \leq i \leq d$, then $s_j \in S$, otherwise, $s_j \notin S$. Regardless of the size of S, the size of $\overrightarrow{B}$ is always d. Thus, the bit-set representation has the advantage of hiding the cardinality of subsets.

The bit-set representation is used to perform the intersection of two data sets S_1 and S_2, namely $S_1 \cap S_2$, with $\overrightarrow{B_1}$ and $\overrightarrow{B_2}$ bit vector representations of size d as follows:

$$\overrightarrow{B_1} * \overrightarrow{B_2} = (b_1^1 \times b_1^2, b_2^1 \times b_2^2, \ldots, b_d^1 \times b_d^2).$$

Also, the cardinality of the set intersection $|S_1 \cap S_2|$ is done by computing the dot product of $\overrightarrow{B_1}$ and $\overrightarrow{B_2}$:

$$\overrightarrow{B_1} \cdot \overrightarrow{B_2} = b_1^1 \times b_1^2 + b_2^1 \times b_2^2 + \cdots + b_d^1 \times b_d^2.$$

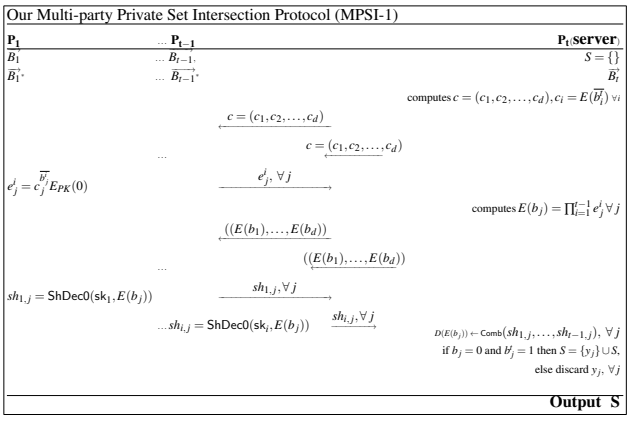

Figure 1: MPSI-1

3.3 The Ruan PSI Protocol

Ruan et al. (Ruan et al., 2019) propose a secure PSI protocol based on the bit-set representation (Section 3.2). In the proposed methodology, two parties P_1 and P_2 own respectively the data sets S_1 and S_2 drawn from a fixed domain $S = (s_1, s_2, \ldots, s_d)$ of d elements. In the setup phase, both parties generate their bit vectors $\overrightarrow{B_1} = (b_1^1, b_2^1, \ldots, b_d^1)$ and $\overrightarrow{B_2} = (b_1^2, b_2^2, \ldots, b_d^2)$ out of their data sets S_1 and S_2, respectively. Party P_2 generates his public/private key pair $(\mathsf{pk}_2, \mathsf{sk}_2)$ and shares pk_2 with P_1. The protocol works as follows:

- P_2 computes $c_j = E_{\mathsf{pk}_2}(b_j^2)$ for all j, $1 \leq j \leq d$ and sends $c = (c_1, c_2, \ldots, c_d)$ to P_1.
- P_1 computes $e_j = (c_j)^{b_j^1} E_{\mathsf{pk}_2}(0)$ for all j, $1 \leq j \leq d$, then sends $e = (e_1, e_2, \ldots, e_d)$ to P_2.
- Upon receiving e, P_2 decrypts every e_j, namely $b_j = D_{\mathsf{sk}_2}(e_j)$, and decides if $b_j = 1$, then $s_j \in S = S_1 \cap S_2$, else $s_j \notin S = S_1 \cap S_2$.

We refer the reader to (Ruan et al., 2019) for details.

4 MPSI-1

We extend the Ruan's PSI protocol into the multi-party setting. To this end, let assume there are t parties $\mathcal{P} = \{P_1, \ldots, P_t\}$ each holding a private data set S_i are interested in obtaining the intersection of their data sets. As previously, the data sets contain n_i number of elements, where $1 \leq i \leq t$. Among all parties, P_t is the server who computes the intersection and the remaining parties are the clients. Differently from the two-party setting, in the multi-party setting we make use of secret shares to protect the privacy of clients so that the server cannot decrypt himself to find mutual intersection of clients with his own data set.

Let pk be the public key of a secret key sk which is shared among $t-1$ clients. The encryption is done

by the threshold Paillier PKE scheme (Fouque et al., 2000)[1]. The protocol execution goes as follows:

- Each party P_i, where $1 \leq i \leq t$, generates the vector representation $\overrightarrow{B_i}$ of their data set S_i.
- Each client P_i $(1 \leq i \leq t-1)$ inverts his bit vectors $\overrightarrow{B_i}^* = (\overline{b_1^i}, \overline{b_2^i}, \ldots, \overline{b_d^i})$.
- P_t then encrypts his bit vector $\overrightarrow{B_t}$ with pk by the threshold Paillier PKE scheme and sends $(c_1, c_2, \ldots, c_d)$ to each client P_i $(1 \leq i \leq t-1)$.
- Each party P_i computes $e^i = (e_1^i, e_2^i \ldots, e_d^i)$ by $e_j^i = c_j^{\overline{b_j^i}} E(0)$ and then sends e^i to P_t.
- Upon receiving e^i's, the server P_t computes $E(b_j) = \prod_{i=1}^{t-1} e_j^i$ and asks ℓ participants among $t-1$ to run $\mathsf{ShDec0}$ protocol to generate their shares $sh_{i,j} = \mathsf{ShDec0}(\mathsf{sk}_i, E(b_j))$ for all j $(1 \leq j \leq n)$. Then, they send their shares $sh_{i,j}$'s to P_t.
- P_t combines these shares by $D(E(b_j)) \leftarrow \mathsf{Comb}(sh_{1,j}, \ldots, sh_{t-1,j})$, $\forall j$, $1 \leq j \leq d$.
- Finally, P_t computes the intersection S by checking: if $b_j = 0$ and $b_j' = 1$ both hold, then $s_j \in S$, otherwise $s_j \notin S$.

Figure 1 summarizes the communications among clients and server for executing the MPSI-1 protocol.

4.1 Correctness

We start by expanding $E(b_j)$ as

$$\begin{aligned} E(b_j) = \prod_{i=1}^{t-1} e_j^i &= c_j^{\overline{b_j^1}} \times c_j^{\overline{b_j^2}} \times \cdots \times c_j^{\overline{b_j^{t-1}}} E(0)^{t-1} \\ &= E(b_j')^{\overline{b_j^1} + \overline{b_j^2} + \cdots + \overline{b_j^{t-1}}} E(0)^{t-1} \\ &= E(b_j' \times (\overline{b_j^2} + \cdots + \overline{b_j^{t-1}}) + 0). \end{aligned}$$

It can be seen that when $b_j' = 0$, P_t always gets $b_j = 0$ and so he concludes that s_j is not in the intersection S. Furthermore, if $b_j' = 1$ and $b_j = r$, then at least one client does not have s_j. Hence, P_t again decides that s_j is not in S. Only, the element s_j is in S, when $b_j' = 1$ and $b_j = 0$ both hold.

4.2 Complexity Anaylsis

Communication Complexity The server P_t sends d ciphertexts to each client in the first, the third, and during the execution of $\mathsf{ShDec0}$. Hence, P_t sends $O(dt)$ ciphertexts in total. Similarly, each client sends $O(d)$ ciphertexts in total. To compare the communication complexity in bits with the previous works, we multiply the number of ciphertexts with $\log|X|$ which is the size of the ciphertext X in bits. Therefore, the communication complexities of P_t and each client in terms of the number of bits sent are $O(dt \log|X|)$ and $O(d \log|X|)$, respectively.

Computational Complexity The server P_t makes d encryptions in the first round. Then, he makes $O(dt)$ multiplications in the third round. Each client performs $O(d)$ encryptions and exponentiations in round two. During the execution of the decryt-to-zero protocol which essentially consists of two parts, each client first randomizes the ciphertext by raising it to the power of a random value and sends back to the server. The server multiplies all randomized ciphertexts performing $O(t)$ multiplications for each element and sends back to the clients. Then, each client calculates his decryption share. This requires $O(d)$ exponentiation for each client. The server makes $O(d\ell)$ multiplications in total with the last round.

4.3 Security Analysis

The privacy of clients is protected, as when $b_j' = 0$ (that is $s_j \notin S_t$), the server P_t can not learn any information of the clients' private data sets. When, $b_j' = 1$, due to $\mathsf{ShDec0}$ protocol, unless $b_j^i = 0$ for all $1 \leq i \leq t-1$ simultaneously, the decryption result will be random. So, there is no information leakage to P_t.

The security of the protocol is based on the assumption of the IND-TCPA (Fouque and Pointcheval, 2001) security of additively homomorphic threshold PKE scheme $T\Pi$ with threshold $\ell < t$. A corrupted party is assumed to be an adversary who is curious but honest. We shall denote $I = \{i_1, \ldots, i_\omega\} \subset \{1, 2, \ldots, t-1\}$ and the complement of I by $\bar{I}$. Let the inputs of the clients be $\mathsf{Inp}_i = (S_i, \mathsf{pk}, \mathsf{sk}_i)$, $i \in \{1, \ldots, t-1\}$ and of the server be $\mathsf{Inp}_t = (S_t, \mathsf{pk})$. The output of the protocol is only provided by the server which is the intersection $S = \bigcap_{j=1}^{t} S_j$. The clients produce no output. We consider two scenarios; (1) the server P_t is honest and a subset P_I of the clients are corrupted, where $\omega < \ell$; (2) the server P_t is corrupted a subset P_I of the clients are corrupted, where $\omega < \ell$.

In the first scenario, where a number of $(\omega < \ell)$ clients are corrupted, the simulator $\mathcal{S}$ has Inp_I but no output (as server is honest). $\mathcal{S}$ is going to simulate the view of corrupted parties (clients) which is indistinguishable from the real view of the protocol. Namely, the simulator will make simulated $\tilde{c} = (\tilde{c}_1, \tilde{c}_2, \ldots, \tilde{c}_d)$ indistinguishable from the real string $c = (c_1, c_2, \ldots, c_d)$. The simulator $\mathcal{S}$ chooses a random binary vector of length d for each honest parties' set, namely the simulated server set B_t and the sets of honest clients $B_{\bar{I}}$s. $\mathcal{S}$ follows the protocol and generates $(\tilde{c}_1, \tilde{c}_2, \ldots, \tilde{c}_d)$. Suppose that there is an adver-

[1]Note that any additively homomorphic threshold PKE scheme can also be used.

sary $\mathcal{A}$ who can distinguish two strings c and $\tilde{c}$ with a non-negligible probability. This implies $\mathcal{A}$ can break IND-TCPA security of Homomorphic PKE. This contradicts the security of the PKE scheme. In addition, other inputs to the corrupted clients are $E(b_j)$'s for $1 \leq j \leq d$. By following the protocol, $\mathcal{S}$ then generates $E(\tilde{b}_j)$'s. As previously, the security of $T\Pi$ does not allow to distinguish $E(b_j)$ and $E(\tilde{b}_j)$ with a non-negligible probability.

In the second scenario, apart from a subset P_I of corrupted clients, the server is also corrupted, where number of corrupted clients ω is less than ℓ. Note that we do not count the server in the number of ℓ as he does not have a secret key share. As the server is among the corrupted parties, the corrupted parties know the intersection $S = \bigcap_{j=1}^{t} S_j$. Hence, the simulator $\mathcal{S}$ has to simulate the output S of the protocol, too. $\mathcal{S}$ starts by choosing random strings for the input set of the honest parties $\tilde{B}_i$, $i \in \bar{I}$ in such a way that $\bigcap_{j\in I} S_j \cap \bigcap_{j\in\bar{I}} \tilde{S}_j = S$. Then, $\mathcal{S}$ follows the protocol and produces simulated $(\tilde{c}_1, \tilde{c}_2, \ldots, \tilde{c}_{n_t})$. Then, he generates $\tilde{e} = (\tilde{e}_1, \tilde{e}_2, \ldots, \tilde{e}_d)$ for the honest parties and corrupted parties. Later, $\mathcal{S}$ generates $E(\tilde{b}_j)$'s for $1 \leq j \leq d$ and sends it to a subset of ℓ parties among t parties. As for the number of corrupted parties we have $\omega < \ell$, there is at least one honest client in this set who receives $E(\tilde{b}_j)$'s. Now, the simulator has to simulate the decryption shares of honest parties without knowing their secret key shares as follows: The simulator knows there are $\ell - \omega$ honest parties and simulates the decryption shares of the honest parties by invokes the share decryption algorithm $\mathsf{ShDec0}$ on $E(\tilde{b}_j)$ and random element from the secret key space, $\tilde{\mathsf{sk}}_k$ for each of $\ell - \omega - 1$ honest parties. For the last remaining honest party, if $y_j \in S$, the simulator computes the decryption share from the decryption shares of the corrupted parties and the simulated honest parties, such that the combining algorithm Comb outputs 0. If $y_j \notin S$, it forces the output to a random value that is a result of the randomness in data. Now clearly the simulated $\tilde{e}$'s are indistinguishable from the real ones, otherwise the IND$-$TCPA security of the TPKE breaks. Also, the decryption shares are produced by a simulator for $\mathsf{ShDec0}$, so the adversary can not distinguish them from the real execution. Note that such a simulator exists for threshold Pailler PKE.

5 MPSI-2

In this section, we propose a similar protocol which is slightly more efficient protocol than MPSI-1. That is, in MPSI-2 we could reduce the server's communication complexity from $O(dt)$ to $O(d\ell)$, where $\ell < t$. The reason is that, the server interacts with the clients only in the shared decryption phase. The key set-up phase is same as in Sec. 4. The protocol follows the steps below:

- Each party P_i, $1 \leq i \leq t$, generates the vector $\overrightarrow{B_i}$ of their data set S_i and inverts it to $\overrightarrow{B_i}^*$.
- Every party P_i, $1 \leq i \leq t$, computes the encryption of their inverted vectors $E(\overrightarrow{B_i}^*) = (E(\overline{b_1^i}), \ldots, E(\overline{b_n^i}))$ by a threshold homomorphic PKE scheme.
- Clients send their encrypted inverted binary vectors $E(\overrightarrow{B_i}^*)$, $1 \leq i \leq t-1$, to the server P_t.
- P_t then makes entry-wise homomorphic addition to obtain
  $(E(b_1), \ldots, E(b_n)) =$
  $(E(\overline{b_1^1} + \overline{b_1^2} + \cdots + \overline{b_1^t}), \ldots, E(\overline{b_n^1} + \overline{b_n^2} \cdots + \overline{b_n^t}))$.
- The server sends $(E(b_1), E(b_2), \ldots, E(b_d))$ to ℓ parties among $t-1$ clients and asks them to mutually decryption-to-zero each $E(b_j)$[2]. We allow $\ell \leq t-1$ for more flexibility, although often $\ell = t-1$.
- Each client P_i involved in the decryption computes their decryption share $sh_{i,j}$ for all $j \in \{1, \ldots, d\}$ as follows: They first send a randomized $E(b_j)^{r_{i,j}}$ to the server; the server combines them into $E(\tilde{b}_j) = E(b_j)^{\Sigma r_{i,j}}$ and sends $E(\tilde{b}_j)$ to each party. Now each party decrypts it to obtain their shares $sh_{i,j}$ for all $j \in \{1, \ldots, d\}$. Finally the shares are sent to the server.[3]
- For each $j \in \{1, \ldots, d\}$, the server runs the combining algorithm on the obtained shares and computes $\mathsf{Dec}(E(\tilde{b}_j)) \leftarrow \mathsf{Comb}(sh_{1,j}, \ldots, sh_{t-1,j})$.
- For each $j \in \{1, \ldots, d\}$, if $\mathsf{Dec}(E(\tilde{b}_j)) = 0$, then the server makes corresponding $c_j = 1$, otherwise $c_j = 0$. Here, $(c_1, c_2, \ldots, c_d)$ is the bit-set representation of the intersection set $S = \cap_i^d S_i$.

5.1 Correctness

The protocol is correct, because when $\overline{b_j^1} + \overline{b_j^2} + \cdots + \overline{b_j^t} = 0$ holds, this means that all $\overline{b_j^t}$'s are zero at the same time. As this bits are inverted, this implies b_j^i's are all one. Therefore, we set $c_j = 1$, and the element s_j corresponding $c_j = 1$ is in the intersection $S = \cap_i^d S_i$. For the case where $\overline{b_j^1} + \overline{b_j^2} + \cdots + \overline{b_j^t} \neq 0$ (which let the $\mathsf{ShDec0}$ output a random value), at least

[2] Note that by randomizing bit vectors of all parties, we have the same functionality of $\mathsf{ShDec0}$.

[3] Note that in Figure 1 we have modified the notations to improve the readability, and this whole step of producing shares with the $\mathsf{ShDec0}$ algorithm is denoted as $sh_{i,j} = \mathsf{ShDec0}(\mathsf{sk}_i, E(b_j))$ for all $j \in \{1, \ldots, d\}$.

one data set does not contain the element corresponding to j-th element in the domain. Hence, we set $c_j = 0$, and the element s_j corresponding $c_j = 0$ is not in the intersection. The protocol works for all data sets, so the error probability of MPSI-2 is zero.

5.2 Complexity Analysis

Communication Complexity The server P_t sends d ciphertexts to ℓ clients among $t-1$ in the second and the fourth rounds (during the execution of ShDec0). Hence, P_t sends $O(d\ell \log|X|)$ bits in total, where $\log|X|$ is the size of the ciphertext X in bits. Likewise, each client sends $O(d \log|X|)$ bits in total.

Computational Complexity Each client and the server perform $O(d)$ encryptions in the first round. Then, in the second round, P_t performs dt homomorphic addition, which corresponds to the modular multiplication in Paillier PKE scheme. During the execution of the decrypt-to-zero protocol, each client first randomizes the ciphertext by raising it to the power of a random value and sends back to the server. The server multiplies all randomized ciphertexts performing $O(d\ell)$ multiplications and re-sends to the clients. Then, each client calculates his decryption share. This requires $O(d)$ exponentiation for each client. The server makes $O(d\ell)$ multiplications in total with the last round. Hence, the dominated complexity for each client and the server is $O(d)$ encryptions.

Note that the protocol is still applicable when the server computes the set intersection and does not have a data set. This leads to that the server does not make any encryption, which reduces the computational overhead for the server significantly.

5.3 Security Analysis

The security of MPSI-2 protocol is guaranteed based on the IND-TCPA security of additively homomorphic threshold PKE scheme $T\Pi$ with threshold $\ell < t$ (Fouque and Pointcheval, 2001). As the steps of the protocols are similar, the security proof of MPSI-1 is can similarly be adopted for MPSI-2. Therefore, to avoid repetitions and save space, we refer readers to Section 4.3 for the proof of this protocol.

5.4 Extension to Other Set Operations

There are three parts in the MPSI protocol presented at the start of Section 5 that can be altered to allow the same scheme to perform different set operations: Whether the bit-set encoding is inverted or not, what the server does after aggregating the bit-sets (in the MPSI protocol the server chooses which bits to decrypt-to-zero at this point) and how the server extracts the final bit-set from the results of the decryption-to-zero. Some of these protocols require the use of a multi-party Secure Comparison Protocol (SCP) such as that by (Kerschbaum et al., 2009) or a Secure Equality Protocol (SEP). The protocols can also be composed for more complex operations by replacing the intermediate decrypt-to-zero with a SEP.

Set Union (MPSU). Multi-party Private Set Union (MPSU) is the process of finding the elements that appeared in at least one of the data sets owned by the participating parties. More precisely, let assume t parties $P_1, \ldots, P_t$ own the data sets $S_{P_1}, \ldots, S_{P_t}$, respectively, are interested in finding $S_{P_1} \cup \ldots \cup S_{P_t}$ without revealing their data sets. Now, we explain how to design MPSU based on MPSI-2 protocol as follows:

- Clients do not invert the bit-sets (*i.e.*, if x_i belongs to S_j, then the associated element in the bit-set is equal to 1) before sending them to the server.
- Aggregation is the same as before, but the entire aggregated bit-set is chosen for a collaborative decryption-to-zero.
- Finally, the server keeps every bit that results as 0 stays 0, and sets every other value to 1. The bits of the aggregated bit-set that equal 1 are the ones that belong to at least one data set, *i.e.*, to set union.

Threshold MPSI (T-MPSI). The Threshold Multi-party Private Set Intersection (T-MPSI) is the process of finding the elements that appeared in the combined sets of parties at least T times. Based on the MPSU protocol, it is as follows:

- The first step is the same as the first step in MPSU protocol, *i.e.*, the bits remain non-inverted.
- Before decryption-to-zero (in the second step of MPSU), the secure comparison protocol (SCP) is used to understand if $b_j \geq$ T for every bit b_j.
- Finally, every bit that results as 0 and every other value becomes 1. The bits which are equivalent to 1 are the ones appeared in more than T data sets.

The server can choose to recover only those elements that they themselves possess by only submitting those bits for decryption, or they can choose to reveal all elements surpassing the threshold by submitting the full bit-set.

Private Set Cardinality (CA). The cardinality is the number of elements in a set S, and it is generally denoted as $|S|$. In some applications it is only necessary to compute the cardinality of a set without knowing which specific elements are inside it. For instance, to privately estimate the number of users

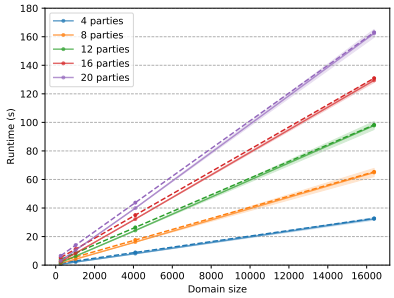

Figure 2: Our runtime over a growing domain size and different number of parties. $n = 16$ for the solid line and $n = 128$ for the dashed line. $\ell = \lfloor \frac{t}{2} \rfloor$.

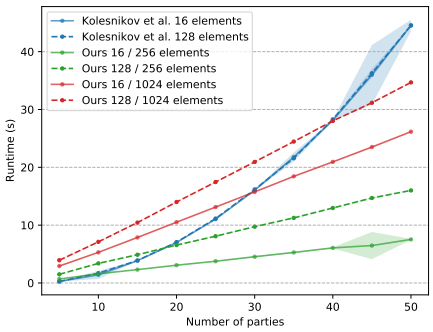

Figure 3: Runtime of our protocol and Kolesnikov et al.'s protocol for $n = 16, 128$, $d = 256, 1024$, and $\ell = \lfloor \frac{t}{2} \rfloor$.

in the Tor network. (Fenske et al., 2017) To extend our protocol for this purpose, the decryption-to-zero is replaced by secure equality protocol (SEP), and for the MPSI protocol, SEP evaluates whether $b_j = 0$ or not, while for the MPSU and T-MPSI, SEP evaluates $b_j = 1$ or not. After that, all bits can be summed homomorphically to reveal the resulting set's cardinality.

Set Subtraction Given two data sets S_A and S_B, the subtraction of S_B from S_A, denoted by $S_A - S_B$ consists of elements that are in S_A but not in S_B. To compute the subtraction of two encrypted bit-sets, the first one should be inverted, but, the second one remains unchanged. After aggregating the bit-sets, a decryption-to-zero is applied on the resulting bit-set. Then, the server changes the bits that are 0 to 1, while the other bits become 0. The bits with value 1 show the elements that belong to the first set but not the second.

6 Results

We have developed a reference implementation[4] of both protocols in C++. The implementation relies on the GMP and NTL libraries. Although the code runs on one machine, it spawns one thread for each client for concurrency; the main thread represents the server.

Set-up We evaluate the runtime of the protocol by performing 10 set intersections for each choice of parameters, and all sets contain n random elements. Our benchmarks were executed on a 64-bit Unix machine with an INTEL CORE™ I7-1065G7 processor at 8 $\times$ 1.30GHz and 16GB of memory. For our work, we choose $\kappa = 1024$ as is common for public-key encryption. We determine the standard deviation σ, and plot the 3σ confidence interval as a shaded area.

Protocol performance As the domain size d is a dominant factor in computation and communication costs (Table 1) of our protocols, we evaluate the runtime over a growing domain size in Figure 2. The collusion threshold is $\ell = \lfloor \frac{t}{2} \rfloor$. The solid and dashed lines represent are when a party owns $n = 16$ and $n = 128$ elements, respectively. The number of parties t in the figure are from the set $\{4, 8, 12, 16, 20\}$. It can be inferred that the runtime is linearly dependent on domain size. The negligible distance between the solid and dashed lines show that the number of elements has negligible impact on the computation cost. For small domain sizes (*i.e* $d \leq 4000$), the growth in the number of parties has small effect on runtime. However, as the domain size grows, the difference between the runtime of different number of parties increases.

We compare the runtime of our MPSI protocol with the protocol proposed by (Kolesnikov et al., 2017) (with $\kappa = 128$) as its empirical runtime is generally superior to that of similar works. Note that their approach is efficient in the number of elements n, but it scales quadratically in terms of the number of parties t. Asymptotically, the computation and communication in our approach is dominated by the domain size d, but for lower values the number of parties t and number of elements n have a linear impact on runtime.

Figure 3 compares the runtime of our implementing proposed approach with Kolesnikov et al.'s approach for $n = 16, 128$, $d = 256, 1024$, and $\ell = \lfloor \frac{t}{2} \rfloor$. Note that for a small number of elements and small domain ($n \leq 16, d \leq 256$), our protocol outperforms the other protocol consistently for any number of parties. However, when the number of elements in each party's set increases, our protocol is more strongly affected than the other work. Still, for many parties, our MPSI protocol becomes considerably more efficient.

In the following table we highlight the parameters for which our MPSI protocol is more efficient than Kolesnikov's protocol. In short, for any domain size d, number of elements n and collusion threshold ℓ, our MPSI protocol is more efficient than Kolesnikov et al.'s protocol when the number of parties is large

[4] https://github.com/jellevos/threshold-multiparty-psi

	$d = 2^8$	$d = 2^{10}$
$n = 16$	$t \geq 11$	$t \geq 29$
$n = 128$	$t \geq 19$	$t \geq 40$

enough. For a domain size of 2^8 elements, regardless of the number of elements in each party's set, the total runtime stays within 16 seconds for a group of 50 parties. For a larger domain of 2^{10} elements, the total runtime stays within 35 seconds for such a group. As expected our protocol is especially efficient for small domains $d \approx 2^8$ and a low number of owned elements $n \approx 16$, where our runtime is significantly lower than the other work, starting from 11 parties.

7 Conclusion

Multi-party Private Set Intersection (MPSI) has been proposed to enable several data owners to find the common elements in their data sets without revealing their data sets. However, the existing solutions suffer from computation and communication costs when the number of parties grows. In this paper, we have proposed a new MPSI approach based on bit-set representation and threshold Paillier PKE, which is efficient for a large number of parties. We show theoretically and empirically that our proposed approach considerably outperforms the existing MPSI solutions when the number of parties increases.

Acknowledgement This work has been supported by the H2020 EU funded project SECREDAS [GA #783119].

REFERENCES

Cao, X., Li, H., Dang, L., and Lin, Y. (2017). A two-party privacy preserving set intersection protocol against malicious users in cloud computing. *Comput. Stand. Interfaces*, 54(P1):41–45.

Cheon, J. H., Jarecki, S., and Seo, J. H. (2012). Multi-Party Privacy-Preserving Set Intersection with Quasi-Linear Complexity. *IEICE Transactions on Fundamentals of Electronics Communications and Computer Sciences*, 95(8):1366–1378.

Dong, C., Chen, L., and Wen, Z. (2013). When private set intersection meets big data: An efficient and scalable protocol. In *the ACM SIGSAC Conference on Computer and Communications Security*, CCS '13, page 789–800.

Fenske, E., Mani, A., Johnson, A., and Sherr, M. (2017). Distributed measurement with private set-union cardinality. In *ACM SIGSAC Conference on Computer and Communications Security, CCS*, pages 2295–2312. ACM.

Fouque, P., Poupard, G., and Stern, J. (2000). Sharing decryption in the context of voting or lotteries. In *Financial Cryptography, 4th International Conference, FC*, volume 1962 of *Lecture Notes in Computer Science*, pages 90–104. Springer.

Fouque, P.-A. and Pointcheval, D. (2001). Threshold cryptosystems secure against chosen-ciphertext attacks. In Boyd, C., editor, *Advances in Cryptology — ASIACRYPT 2001*, pages 351–368, Berlin, Heidelberg. Springer Berlin Heidelberg.

Goldreich, O. and Warning, A. (1998). Secure multi-party computation.

Hazay, C. and Venkitasubramaniam, M. (2017). Scalable multi-party private set-intersection. In Fehr, S., editor, *Public-Key Cryptography (PKC)*, pages 175–203. Springer Berlin Heidelberg.

Inbar, R., Omri, E., and Pinkas, B. (2018). Efficient scalable multiparty private set-intersection via garbled bloom filters. In *11th International Conference on Security and Cryptography for Networks*, pages 235–252.

Kerschbaum, F., Biswas, D., and de Hoogh, S. (2009). Performance comparison of secure comparison protocols. In *Database and Expert Systems Applications, DEXA*, pages 133–136.

Kissner, L. and Song, D. X. (2005). Privacy-preserving set operations. In *Annual International Cryptology Conference (CRYPTO)*, volume 3621 of *Lecture Notes in Computer Science*, pages 241–257. Springer.

Kolesnikov, V., Matania, N., Pinkas, B., Rosulek, M., and Trieu, N. (2017). Practical multi-party private set intersection from symmetric-key techniques. In *ACM SIGSAC Conference on Computer and Communications Security, CCS*, pages 1257–1272. ACM.

Miyaji, A. and Nishida, S. (2015). A scalable multiparty private set intersection. In *Network and System Security Conference, NSS*, volume 9408 of *Lecture Notes in Computer Science*, pages 376–385. Springer.

Ruan, O., Wang, Z., Mi, J., and Zhang, M. (2019). New approach to set representation and practical private set-intersection protocols. *IEEE Access*, 7:64897–64906.

Sheikhalishahi, M., Tillem, G., Erkin, Z., and Zannone, N. (2019). Privacy-preserving multi-party access control. In *ACM Workshop on Privacy in the Electronic Society, WPESCCS*, pages 1–13.