Privacy-Preserving Distributed Set Intersection

Qingsong Ye, Huaxiong Wang and Christophe Tartary

Department of Computing, Macquarie University, Australia
{qingsong,hwang,ctartary}@ics.mq.edu.au

Abstract. With the growing demand of databases outsourcing and its security concerns, we investigate privacy-preserving set intersection in a distributed scenario. We propose a one-round protocol for privacy-preserving set intersection based on a combination of secret sharing scheme and homomorphic encryption. We then show that, with an extra permutation performed by each of contacted servers, the cardinality of set intersection can be computed efficiently. All protocols constructed in this paper are provably secure against a semi-honest adversary under the Decisional Diffie-Hellman assumption.

Keywords: privacy-preserving set intersection, homomorphic encryption

1 Introduction

Privacy-Preserving Set Intersection (PPSI) protocols [7] are cryptographic techniques allowing two or more parties, each holding a set of inputs, to jointly calculate set operations of their inputs without leaking any information to each other. Consider that two companies C_1 and C_2 want to discover the consumption pattern of their shared customers. That is, they want to determine the likelihood that a customer buying the product P_1 from C_1 is also buying the product P_2 from C_2. To obtain this information, they would like to perform a set intersection operation on their private datasets. In order to preserve confidentiality of the companies business and to protect the customers' privacy, the purchase details of customers must not be revealed. There are many other examples of PPSI applications such as when two hospitals conduct a study where they wish to analyze patients records anonymously.

With the growing demand of databases outsourcing and security requirements imposed on its applications, we investigate PPSI in a distributed environment. We call this Privacy-Preserving Distributed Set Intersection (PPDSI). To illustrate the security problem, we consider the following scenario. Assume that a provider owning a dataset wishes to outsource it to commercial servers and make it available to his clients. If he outsources his dataset to a single server then he has to fully trust that server and risk the privacy of his data. Alternatively, he can encrypt his dataset before sending it to the server but querying and evaluating on such encrypted data are very inefficient.

In order to protect the dataset privacy at an acceptable efficiency cost, we could let the provider distribute the dataset to w servers using a (t, w)-threshold secret sharing scheme. As such, any $t-1$ or less servers should not able to find out the original data. Now, assume that a client holding her private dataset, wishes to compute a specific set operation for the two sets held by the provider and herself. In order to do this successfully, the client interacts with t or more servers. In our settings, we require that this interaction is done with minimum possible disclosure of information, that is, the client learns nothing except the final result of the set operation.

In general, PPSI can be implemented using secure multi-party computation protocols [3, 23]. However, such solutions are generally inefficient. More specialized protocols on PPSI are needed to improve its efficiency.

Related Work. A specialized private set intersection protocol recently developed by Freedman, Nissim and Pinkas (FNP) [7] is based on the representation of datasets as roots of a polynomial. To briefly describe Freedman et al.'s construction, suppose $CS = (K, E_{\text{pk}}, D_{\text{sk}})$ is a semantically secure public-key homomorphic encryption scheme. Assume that Alice has the dataset $A = \{a_1, \ldots, a_n\}$ and Bob owns the dataset $B = \{b_1, \ldots, b_m\}$.

To evaluate $A \cap B$, Alice constructs the polynomial $f(x) = \prod_{a_i \in A}(x - a_i) = \sum_{i=0}^{n} \alpha_i x^i$. Then, she encrypts each coefficient as $E_{\text{pk}}(\alpha_i)$ with the homomorphic cryptosystem CS such as Paillier's [18] and the standard variant of the ElGamal encryption scheme (see [4]). Note that an homomorphic cryptosystem allows a party knowing $E_{\text{pk}}(x)$ and $E_{\text{pk}}(y)$ to compute $E_{\text{pk}}(x + y) = E_{\text{pk}}(x) \cdot E_{\text{pk}}(y)$ and $E_{\text{pk}}(x \cdot c) = E_{\text{pk}}(x)^c$ where c is any constant. The reader is referred to Sect. 2.1 for a formal definition. Note that we only use the standard variant of the ElGamal encryption scheme in our protocols due to our distributed setting.

Thus, given encrypted coefficients, Bob can obliviously evaluate $E_{\text{pk}}(f(b_i))$ for each element $b_i \in B$. Note that if $b_i \in A$ then $f(b_i) = 0$. Since Bob does not want to reveal any other information when $b_i \notin A$, he randomizes all his oblivious evaluations by a random nonzero value r as $E_{\text{pk}}(f(b_i))^r = E_{\text{pk}}(r \cdot f(b_i))$. Consequently, if $f(b_i) = 0$ then the encryption of $E_{\text{pk}}(r \cdot f(b_i)) = E_{\text{pk}}(0)$. Otherwise, $E_{\text{pk}}(r \cdot f(b_i))$ is some random value. This hides any information about elements in B which are not in A. To enable Alice to check whether b_i also belongs to her dataset, Bob sends all the cryptograms $E_{\text{pk}}(r \cdot f(b_i) + b_i)$'s to her. She decrypts them and tests whether any of the resulting plaintexts are in A as $D_{\text{sk}}(E_{\text{pk}}(r \cdot f(b_i) + b_i)) = b_i$ if and only if $b_i \in A$.

Inspired by FNP, Kessner and Song (KS) [12] propose a solution to various privacy-preserving set operations such as set union, set intersection, the cardinality of set intersection and multiplicity testing. Based on a threshold homomorphic cryptosystem, Sang et al. gave protocols for the set intersection and set matching problems with an improved computation and communication complexity in [20].

Protocols for testing the subset relation in a two-party setting are also discussed in [11, 13] while the set disjointness test are introduced in [10, 9]. Note that checking the equality of two datasets is a special case of the private disjointness problem, where each party has a single element in the database. Such protocols were considered in [6, 16, 14].

Our Results. Our approach is based on homomorphic encryption and secret sharing. This paper builds on recently developed private set operation protocols FNP and KS and offers a new construction in two-party private set operations where one dataset is distributed.

Contrary to the previous two-party PPSI protocols based on the one client-one server setting, we deal with the distributed case relying on secret sharing described earlier. Our construction may be of great value where the privacy of unencrypted dataset outsourced in single server is a great concern.

We first compute the w shares of the dataset $\mathcal{D}_{\mathcal{P}}$ of the provider $\mathcal{P}$ by constructing a bivariate polynomial and evaluating it at w points to get the shares. This approach is to make the share construction more efficient. Our construction only needs to use Shamir's secret scheme [21] a single time to compute the shares of the whole dataset.

In our set intersection protocol, we will use our observation that $\sum_{j=1}^{t} b_j\,(\mu_j - \mu) = 0$ where the b_j's are Lagrange interpolation coefficients, μ is a value and the μ_j's are the shares of μ. Using this relation, one can obliviously check that an element c is equal to μ by collecting t values $(\mu_j - c)\,r$ where r is a randomizer common to all participants. As a consequence, if the client $\mathcal{C}$ interacts in parallel with t servers with her whole dataset $\mathcal{D}_{\mathcal{C}}$, she is able to compute $\mathcal{D}_{\mathcal{P}} \cap \mathcal{D}_{\mathcal{C}}$.

We then extend our PPDSI solution to a one-round protocol evaluating $|\mathcal{D}_{\mathcal{C}} \cap \mathcal{D}_{\mathcal{P}}|$. To prevent the client from learning the intersection $\mathcal{D}_{\mathcal{C}} \cap \mathcal{D}_{\mathcal{P}}$, each server will permute the cryptograms it has evaluated obliviously before sending them back to the client $\mathcal{C}$. Thus, after decryption and computation, the client only learns $|\mathcal{D}_{\mathcal{C}} \cap \mathcal{D}_{\mathcal{P}}|$.

Our protocols are secure against an honest-but-curious (semi-honest) adversary. By definition, such an adversary follows the steps of the protocol execution but tries to learn extra information from the messages received during its execution. Our homomorphic encryption is based on the ElGamal cryptosystem [5] which is semantically secure provided the Decisional Diffie-Hellman (DDH) assumption holds [22]. The security of this building block will imply the security of our protocols. Note that, as in [7, 10], our protocols reveal the size of the datasets of both the client and the provider. As suggested in [10], "dummy" elements can be used for dataset padding in order to hide the size of the original dataset. But, in this case, the protocols reveal an upper bound on the number of elements in the sets.

The complexity of the communication cost for each of these two constructions is $O(t\,|\mathcal{D}_{\mathcal{P}}|\,|\mathcal{D}_{\mathcal{C}}| \times \log_2 p)$ bits. The computation cost complexity is $O(t\,|\mathcal{D}_{\mathcal{P}}|\,|\mathcal{D}_{\mathcal{C}}| \times \log_2^3 p)$ bits for our two protocols. These complexity results are efficient considering our distributed setting.

Our paper is organized as follows. In Sect. 2, we introduce the cryptographic primitive used in our protocols, describe the distributed environment in which our protocols are run, and give the adversary model. In Sect. 3, we present our two protocols for the set intersection problem and the cardinality of set intersection problem. The security and efficiency of these two schemes are analyzed in that section as well. Finally, in Sect. 4, we give concluding remarks and discuss possible future work.

2 Preliminaries

2.1 Additive Homomorphic Encryption

We will utilize an additive homomorphic public key cryptosystem. Following Adida and Wikstrom [1], we use the following definition.

Definition 1 ([1]). *A cryptosystem with key generator K, encryption algorithm E_{pk} and decryption algorithm D_{sk} is said to be homomorphic if for every key pair $(pk, sk) \in K(1^l)$:*

1. *The message space $\mathcal{M}$ is a subset of an abelian group $\mathcal{G}(\mathcal{M})$ written additively.*
2. *The randomizer space $\mathcal{R}$ is an abelian group written additively.*
3. *The ciphertext space $\mathcal{C}$ is an abelian group written multiplicatively.*
4. *The group operations can be computed in polynomial time given pk. For every $m, m' \in \mathcal{M}$ and $r, r' \in \mathcal{R}$, we have $E_{pk}(m, r) \odot E_{pk}(m', r') = E_{pk}(m+m', r+r')$.*
5. *The cryptosystem is said to be additive if the message space $\mathcal{M}$ is the additive modular group $\mathbb{Z}_n$ for some integer $n > 1$.*

When such operations are performed, we require that the resulting ciphertexts be re-randomized for security. During such a process, the ciphertext c of the plaintext m is transformed into c' such that c' is still a valid cryptogram for the message m but relying on a different random string from c's.

In our protocols, the computations are carried out over $\mathbb{Z}_\mathrm{p}$ where p is prime. We assume that $p = 2\,q + 1$ where q is also prime. We note that all our protocols can be based on the standard variant of the ElGamal encryption scheme (see [4]) which recently was used for constructing privacy-preserving set operation protocols in [10, 2, 14].

Let g, h and f be three random generators of order q in $\mathbb{Z}_\mathrm{p}^*$, $m_1, m_2, m \in \mathbb{Z}_\mathrm{q}$ and corresponding $r_1, r_2, r \overset{\mathrm{R}}{\leftarrow} \mathbb{Z}_\mathrm{q}$. We denote $\odot$ the multiplication over $\mathbb{Z}_\mathrm{p} \times \mathbb{Z}_\mathrm{p}$ defined as follows.

$$E_\mathrm{pk}(r_1, m_1) \odot E_\mathrm{pk}(r_2, m_2) := \left(g^{r_1+r_2}, h^{r_1+r_2} f^{m_1+m_2}\right) = E_\mathrm{pk}(r_1 + r_2, m_1 + m_2)$$

If we repeat this operation c times for a single encryption, then we have

$$E_\mathrm{pk}(r, m)^c = \underbrace{E_\mathrm{pk}(r, m) \odot E_\mathrm{pk}(r, m) \odot \ldots \odot E_\mathrm{pk}(r, m)}_{\text{c times}} := E_\mathrm{pk}(c\,r, c\,m)$$

For simplicity, we use $E_{\text{pk}}(m)$ to represent $E_{\text{pk}}(r, m)$ in the rest of the presentation as we assume that there is always a corresponding $r \xleftarrow{\text{R}} \mathbb{Z}_{\text{q}}$.

2.2 Distributed Environment

The players are a client $\mathcal{C}$, a provider $\mathcal{P}$, and w servers $S_1, S_2, \ldots, S_w$. We assume that the provider holds a dataset $\mathcal{D}_{\mathcal{P}} = \{\mu_0, \mu_1, \ldots, \mu_{n-1}\}$ which is distributed to w servers using (t, w)-Shamir's secret sharing scheme.

The provider $\mathcal{P}$ does not directly interact with $\mathcal{C}$ for the set operations. Instead, $\mathcal{C}$ contacts at least t servers to perform the requested set operation. Note that this distributed setting was first proposed by Naor and Pinkas [17].

Our homomorphic encryption system is based on a variant of ElGamal cryptosystem in which the message space is over $\mathbb{Z}_{\text{q}}$ where $q \geq n$. For simplicity, we omit modulus q within the computation of shares construction in this section.

Initialization and Share Distribution Phase. $\mathcal{P}$ constructs a polynomial $F(y)$ whose coefficients represent his dataset $\mathcal{D}_{\mathcal{P}}$, i.e.:

$$F(y) = \sum_{i=0}^{n-1} \mu_i y^i$$

Then, $\mathcal{P}$ generates a random masking bivariate polynomial $H(x, y)$ as:

$$H(x, y) = \sum_{j=1}^{t-1} \sum_{i=0}^{n-1} a_{j,i} x^j y^i \quad \text{where } a_{j,i} \xleftarrow{\text{R}} \mathbb{Z}_{\text{q}}$$

Note that we have $H(0, y) = 0$ for any y. Using the polynomial $H(x, y)$, $\mathcal{P}$ defines another bivariate polynomial $Q(x, y) = F(y) + H(x, y)$. Note that we get $\forall y \;\; Q(0, y) := F(y)$. For $1 \leq \ell \leq w$, $\mathcal{P}$ sends $Q(\ell, y) := \sum\limits_{i=0}^{n-1} \mu_{i,\ell}\, y^i$ to server S_ℓ where $\forall i \in \{0, \ldots, t-1\}$ $\mu_{i,\ell} = \mu_i + b_{i,\ell}$ with $b_{i,\ell} = \sum\limits_{j=1}^{t-1} a_{j,i}\, \ell^j$. The server S_ℓ receives a set of shared coefficients $\{\mu_{0,\ell}, \;\ldots,\; \mu_{n-1,\ell}\}$ of the polynomial $F(y)$ (see [15]).

Secret Reconstruction Phase. We now show how any t-subset of servers can recover $F(y)$. Denote $S_{\ell_1}, \ldots, S_{\ell_t}$ the t servers contacted by the client. Using Lagrange interpolation formula, we know that the coalition of t or more servers can reconstruct the original polynomial $F(y)$. The t polynomials $Q(\ell_m, y)$ for $m \in [1, \ldots, t]$ verify the following system:

$$V^{-1}\begin{pmatrix} Q(\ell_1, y) \\ Q(\ell_2, y) \\ \vdots \\ Q(\ell_t, y) \end{pmatrix} = \begin{pmatrix} F(y) \\ \sum_{j=1}^{t-1} a_{j,0}\, y^i \\ \vdots \\ \sum_{j=1}^{t-1} a_{j,n-1}\, y^i \end{pmatrix} \tag{1}$$

where V is the $t \times t$ Vandermonde matrix: $V := \left(\ell_i^j\right)_{i=1,\ldots,t}^{j=0,\ldots,t-1}$. Since we are only interested in the reconstruction of $F(y)$, we simply need to know the first row $(v_{1,1} \cdots v_{1,t})$ of V^{-1} as we have:

$$\sum_{j=1}^{t} v_{1,j}\, Q(\ell_j, y) = F(y)$$

As a consequence, we obtain:

$$\forall i \in \{0, \ldots, n-1\} \quad \sum_{j=1}^{t} v_{1,j}\, \mu_{i,\ell_j} = \mu_i \tag{2}$$

Lemma 1 shows how to construct the first row of V^{-1} whose proof can be found in Appendix A.

Lemma 1. *We have:*

$$\left(\forall j \in \{1, \ldots, t\} \quad v_{1,j} = \prod_{\substack{1 \le k \le t \\ k \ne j}} \frac{\ell_k}{\ell_k - \ell_j}\right) \quad \textit{and} \quad \sum_{j=1}^{t} v_{1,j} = 1$$

From the previous lemma, we deduce:

$$\forall i \in \{1, \ldots, n-1\} \quad \mu_i = \sum_{j=1}^{t} \left(\prod_{\substack{1 \le k \le t \\ k \ne j}} \frac{\ell_k}{\ell_k - \ell_j} \right) \mu_{i,\ell_j}$$

Note that our reconstruction technique can be seen as a particular case of Lagrange interpolation. Notice that we use the variant of ElGamal that is defined over $\mathbb{Z}_\mathrm{p}$. Further in our paper, the computations are being done modulo p and we simplify the notation by skipping the modulus in the congruences. If we use different modulus, the congruence will be written in full to avoid confusion.

2.3 Adversary Model

We consider a semi-honest adversary model. Due to space constraints, we only provide the intuition and informal definitions of this model. The reader is referred to [8] for a more complete discussion.

In this model, there is no direct interaction between $\mathcal{C}$ and $\mathcal{P}$. Instead the client $\mathcal{C}$ and w servers are assumed to follow the steps defined in the protocol. The security definition is straightforward that only the client $\mathcal{C}$ learns the result of the protocol.

Definition 2. *(t-secure). A set operation protocol is said to be t-secure if the client $\mathcal{C}$, colluding with at most $t-1$ out of w servers learns no information about the provider's dataset and the set operation result.*

Following [16, 17] our model should meet the following requirements:

Correctness. A protocol is correct if the client $\mathcal{C}$ is able to compute the valid result from shares obtained from t servers assuming that each server and the client honestly follow the protocol.

***Client*'s security.** The protocol should guarantee the client privacy, i.e. the servers learn nothing about either the client inputs or its corresponding computed output. In other words, a server is not able to distinguish the client inputs from uniform random variables.

***Provider*'s security.** The protocol should not give out to the client any information about the function held by the provider apart from the output of the function assuming that no server colludes with the client. Also, the provider privacy is t-secure.

3 Protocols for PPDSI

In this section, we address the problem of designing protocols for PPDSI related issues. Those targeted in this paper are the privacy preserving set intersection problem and the cardinality of set intersection problem. The provider $\mathcal{P}$ holds a dataset $\mathcal{D}_{\mathcal{P}} = \{\mu_0, \ldots, \mu_{n-1}\}$ which is distributed to w servers $S_1, \ldots, S_w$ as in Sect. 2.2. The dataset $\mathcal{D}_{\mathcal{C}}$ of the client $\mathcal{C}$ is $\{c_0, \ldots, c_{m-1}\}$. Note that it is assumed that $|\mathcal{D}_{\mathcal{C}}| = m$ and $|\mathcal{D}_{\mathcal{P}}| = n$ are publicly known. We assume that the provider $\mathcal{P}$ broadcasts $\lambda \xleftarrow{\mathrm{R}} \mathbb{Z}_{\mathrm{q}} - \{0\}$ to the w servers.

3.1 Determination of the Set Intersection

Figure 1 represents a protocol which enables the client $\mathcal{C}$ to compute the intersection $\mathcal{D}_{\mathcal{P}} \cap \mathcal{D}_{\mathcal{C}}$ by contacting any t-subset of servers $S_{\ell_1}, \ldots, S_{\ell_t}$.

Correctness of the Protocol. In order to prove the soundness of our construction, we need the following lemma.

Lemma 2. *Let $b_j = \prod_{\substack{1 \le k \le t \\ k \ne j}} \frac{\ell_k}{\ell_k - \ell_j}$ be the coefficient constructed at Step 3.b of Fig. 1. Then:*

$$\sum_{j=1}^{t} b_j(\mu_{i,\ell_j} - \mu_i) = 0$$

Input: The client $\mathcal{C}$ has a set of data $\mathcal{D}_\mathcal{C}$. Each server $S_\ell (1 \leq \ell \leq w)$ knows the random value λ and the shared coefficients $\{\mu_{0,\ell}, \ldots, \mu_{n-1,\ell}\}$ of the polynomial $F(y)$ (whose coefficients are the elements of the provider's dataset $\mathcal{D}_\mathcal{P}$).
Output: The client $\mathcal{C}$ learns $\mathcal{D}_\mathcal{C} \cap \mathcal{D}_\mathcal{P}$.

1. $\mathcal{C}$ generates a new key pair $(pk, sk) \leftarrow K(1^l)$, $r \xleftarrow{\text{R}} \mathbb{Z}$q, then
   (a) sets $\tau_\upsilon \leftarrow E_{\text{pk}}(c_\upsilon) = (g^r, h^r f^{-c_\upsilon})$ for $\upsilon \in [0, \ldots, m-1]$.
   (b) broadcasts $\{\ pk, \tau_0, \ldots, \tau_{m-1}\ \}$ to t servers $S_{\ell_1}, \ldots, S_{\ell_t}$.
2. For $j = 1, \ldots, t$ each contacted server S_{ℓ_j}
   (a) computes $\tau_{\upsilon,i,j} \leftarrow (g^{\lambda r}, h^{\lambda r} f^{\lambda(\mu_{i,\ell_j} - c_\upsilon)})$ for $\upsilon \in [0, \ldots, m-1]$ and $i \in [0, \ldots, n-1]$.
   (b) sends $\{\tau_{0,0,j},\ \ldots,\ \tau_{0,n-1,j},\ \tau_{1,0,j},\ \ldots,\ \tau_{m-1,n-1,j}\ \}$ to $\mathcal{C}$.
3. For $\upsilon = 0, \ldots, m-1$, the client $\mathcal{C}$
   (a) computes $d_{\upsilon,i,j} \leftarrow D_{\text{sk}}(\tau_{\upsilon,i,j})$ for $i \in [0, \ldots, n-1]$, $j \in [1, \ldots, t]$.
   (b) computes $d_{\upsilon,i} \leftarrow \prod_{j=1}^{t} (d_{\upsilon,i,j})^{b_j}$ for $i = 0, \ldots, n-1$, where $b_j = \prod_{\substack{1 \leq k \leq t \\ k \neq j}} \frac{\ell_k}{\ell_k - \ell_j}$ in the Lagrange interpolation formula.
   (c) concludes $c_\upsilon \in \mathcal{D}_\mathcal{P}$, if $d_{\upsilon,i} = 1$ for $i \in [0, \ldots, n-1]$; otherwise $d_{\upsilon,i}$ is a random integer.
4. When this process concludes, $\mathcal{C}$ learns $\mathcal{D}_\mathcal{C} \cap \mathcal{D}_\mathcal{P}$.

Fig. 1. Privacy-preserving set intersection protocol

Proof. Note that our coefficient b_j corresponds to the j^{th} coefficient of the first row of the matrix V^{-1} denoted $v_{1,j}$ in Sect. 2.2. From Lemma 1, we get that $\sum_{j=1}^{t} b_j = 1$. Thus, (2) provides our result. □

In the above protocol, the client $\mathcal{C}$ first encrypts each element c_υ of her dataset by using her public key as $E_{\text{pk}}(f^{-c_\upsilon})$ for $\upsilon \in [0, \ldots, m-1]$ and broadcasts all these encrypted elements to t servers. For each encrypted element $E_{\text{pk}}(f^{-c_\upsilon})$, the servers S_{ℓ_j} $(1 \leq j \leq t)$ compute $E_{\text{pk}}(f^{\lambda(\mu_{i,\ell_j} - c_\upsilon)})$ for $i \in [0, \ldots, n-1]$, and send all the $E_{\text{pk}}(f^{\lambda(\mu_{i,\ell_j} - c_\upsilon)})$'s back to $\mathcal{C}$. The client $\mathcal{C}$ then decrypts those $E_{\text{pk}}(f^{\lambda(\mu_{i,\ell_j} - c_\upsilon)})$'s and computes $f^{\lambda \sum_{j=1}^{t} b_j\ (\mu_{i,\ell_j} - c_\upsilon)}$ for each $i = 0, \ldots, n-1$. Note that if $c_\upsilon = \mu_i$ then $\sum_{j=1}^{t} b_j\ (\mu_{i,\ell_j} - c_\upsilon) = 0$. Therefore, the client $\mathcal{C}$ learns that $c_\upsilon \in \mathcal{D}_{\text{P}}$ if there exists $i \in [0, \ldots, n-1]$ such that $f^{\lambda \sum_{j=1}^{t} b_j\ (\mu_{i,\ell_j} - c_\upsilon)} = 1$. When all the steps are finished, $\mathcal{C}$ learns $\mathcal{D}_{\text{C}} \cap \mathcal{D}_{\text{P}}$.

Security of the Construction. The two theorems given below characterize the security of the set intersection protocol. Their proofs can be found in Appendix B and C.

Theorem 1. *Given the set intersection protocol described in Fig. 1 and assuming that the underlying homomorphic encryption is semantically secure, then each of the contacted servers cannot distinguish inputs generated by the client $\mathcal{C}$ from random integers with a non-negligible probability.*

Theorem 2. *Assuming that the discrete logarithm problem is hard, the client $\mathcal{C}$ cannot compute any information about shared coefficients $\{\mu_{0,\ell}, \ldots, \mu_{n-1,\ell}\}$ $(1 \leq \ell \leq w)$ distributed by the provider $\mathcal{P}$. In addition, $\mathcal{P}$'s privacy is t-secure.*

3.2 Computation of the Cardinality of Set-Intersection

By introducing a permutation into our PPDSI protocol, we develop an algorithm computing the cardinality of the datasets' intersection $|\mathcal{D}_\mathcal{C} \cap \mathcal{D}_\mathcal{P}|$. Denote $\mathbb{P}_{m\,n}$ the set of all permutations of $\{1, \ldots, m\,n\}$. Assume that $\mathcal{P}$ has a private permutation function π, chosen uniformly at random from $\mathbb{P}_{m\,n}$, which is given to the w servers. This scheme is represented as Fig. 2.

Input: The client $\mathcal{C}$ has a set of data $\mathcal{D}_\mathcal{C}$. Each server $S_\ell (1 \leq \ell \leq w)$ knows the random value λ, the permutation function π and the shared coefficients $\{\mu_{0,\ell}, \ldots, \mu_{n-1,\ell}\}$ of the polynomial $F(y)$ (whose coefficients are the elements of the provider's dataset $\mathcal{D}_\mathcal{P}$).
Output: The client $\mathcal{C}$ learns $|\mathcal{D}_\mathcal{C} \cap \mathcal{D}_\mathcal{P}|$.

1. $\mathcal{C}$ generates a new key pair $(pk, sk) \leftarrow K(1^l), r \xleftarrow{\text{R}} \mathbb{Z}_\text{q}$, then
   (a) sets $\tau_\upsilon \leftarrow E_\text{pk}(c_\upsilon) = (g^r, h^r f^{-c_\upsilon})$ for $\upsilon \in [0, \ldots, m-1]$.
   (b) broadcasts $\{\ pk, \tau_0, \ldots, \tau_{m-1}\ \}$ to t servers $S_{\ell_1}, \ldots, S_{\ell_t}$.
2. For $j = 1, \ldots, t$ each contacted server S_{ℓ_j}
   (a) computes $\tau_{\upsilon,i,j} \leftarrow (g^{\lambda r}, h^{\lambda r} f^{\lambda(\mu_{i,\ell_j} - c_\upsilon)})$ for $\upsilon \in [0, \ldots, m-1]$ and $i \in [0, \ldots, n-1]$.
   (b) obtains $\{\tau_{\pi(0,0),j},\ \ldots,\ \tau_{\pi(0,n-1),j},\ \tau_{\pi(1,0),j},\ \ldots,\ \tau_{\pi(m-1,n-1),j}\} \leftarrow \pi(\tau_{0,0,j},\ \ldots,\ \tau_{0,n-1,j},\ \tau_{1,0,j},\ \ldots,\ \tau_{m-1,n-1,j})$
   (c) sends $\{\tau_{\pi(0,0),j},\ \ldots,\ \tau_{\pi(0,n-1),j},\ \tau_{\pi(1,0),j},\ \ldots,\ \tau_{\pi(m-1,n-1),j}\}$ to $\mathcal{C}$.
3. For $\upsilon' = 0, \ldots, m-1$, the client $\mathcal{C}$
   (a) computes $d_{\pi(\upsilon',i),j} \leftarrow D_\text{sk}(\tau_{\pi(\upsilon',i),j})$ for $i \in [0, \ldots, n-1]$, $j \in [1, \ldots, t]$.
   (b) computes $d_{\pi(\upsilon',i)} \leftarrow \prod_{j=1}^{t} \left(d_{\pi(\upsilon',i),j}\right)^{b_j}$ for $i \in [0, \ldots, n-1]$, where $b_j = \prod_{\substack{1 \leq k \leq t \\ k \neq j}} \frac{\ell_k}{\ell_k - \ell_j}$ in the Lagrange interpolation formula.
4. When this process concludes, $\mathcal{C}$ learns $|\mathcal{D}_\mathcal{C} \cap \mathcal{D}_\mathcal{P}|$ as it is the number of $d_{\pi(\upsilon',i)}$'s equal to 1.

Fig. 2. Privacy-preserving cardinality of set-intersection protocol

The protocol, given in Fig. 2, works in the same way as the distributed set intersection protocol with the addition that all servers run the same permutation function

π on their computed cryptograms. This is to prevent the client $\mathcal{C}$ from learning the set intersection $\mathcal{D}_{\mathcal{C}} \cap \mathcal{D}_{\mathcal{P}}$.

Security of the Construction. The security model as well as the proof for this protocol are similar to what was done for our set-intersection protocol presented in Sect. 3.1 as the permutation π was chosen uniformly at random from $\mathbb{P}_{m\,n}$.

3.3 Efficiency of our Protocols

In this part, we study the communication and computation cost of our two constructions.

Communication Cost. For both protocols, $\mathcal{C}$ broadcasts a set of m encrypted values to t servers while each contacted server S_{ℓ_j} responds with $m\,n$ messages. Thus, the complexity of the communication cost for both constructions are $O(t\,m\,n \times \log_2 p)$ bits.

Computation Cost. It should be noticed that operations in $\mathbb{Z}_{\mathrm{p}}$ can be done in $O(\log_2^2 p)$ bit operations.

For our first protocol, $\mathcal{C}$ needs $m+2$ modular exponentiations and m modular multiplications to encrypt her dataset, $t\,m\,n$ decryptions, $t\,m\,n$ modular exponentiations and $m\,n\,(t-1)$ modular multiplications for Lagrange interpolation. Note that each decryption represents one modular multiplication and one modular exponentiation. Each server S_{ℓ_j} executes $m\,n$ modular exponentiations and multiplications when processing its shares. So, this protocol uses $O(t\,m\,n \times \log_2 p)$ modular multiplications considering that a single modular exponentiation takes at most $\lfloor \log_2(p-1) \rfloor$ modular multiplications using the Fast Exponentiation algorithm presented in [19].

The cost of our second protocol is the same as the first one's plus t executions of the permutation π. Assuming that π is represented by its binary permutation matrix M_π, each of these t queries has a negligible cost since π is a simple reordering of its inputs (M_π has a single coefficient equal to 1 per row).

Therefore, the complexity of computation cost of these two constructions is $O(t\,m\,n \times \log_2^3 p)$ bits.

4 Conclusion and Future Work

In this paper, we have proposed a protocol for the privacy-preserving set intersection computation in a distributed environment. Our construction was based on Shamir's secret sharing scheme and homomorphic encryption scheme. With our construction, each server only held the shares of the original provider dataset, and consequently the privacy of that dataset was protected. Moreover, we have shown that, using a permutation π, we could efficiently compute the cardinality of the set intersection.

Further research will be to focus on providing a solution of the above distributed set intersection and the cardinality of set intersection problems against an active adversary.

References

[1] B. Adida and D. Wikstrom. How to shuffle in public. In *4th Theory of Cryptography Conference (TCC '07),*, volume accepted of *LNCS*. Springer-Verlag, 2007.

[2] B. Aiello, Y. Ishai, and O. Reingold. Priced oblivious transfer: How to sell digital goods. In B. Pfitzmann, editor, *Advances in Cryptology - Eurocrypt '01*, volume 2045 of *LNCS*, pages 119–135. Springer-Verlag Berlin Heidelberg, 2001.

[3] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *20th Annual ACM Symposium on Theory of Computing*, pages 1–10. ACM Press, 1988.

[4] R. Cramer, R. Gennaro, and B. Schoenmakers. A secure and optimally efficient multi-authority election scheme. In *Advances in Cryptology - Eurocrypt'97*, volume 1233 of *LNCS*, pages 103 – 118. Springer - Verlag, 1997.

[5] T. El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In G. R. Blakley and D. Chaum, editors, *Advances in Cryptology - Crypto '84*, volume 196, pages 19–22. Springer-Verlag, August 1984.

[6] R. Fagin, M. Naor, and P. Winkler. Comparing information without leaking it. *Communications of the ACM*, 39(5):77–85, 1996.

[7] M. J. Freedman, K. Nissim, and B. Pinkas. Efficient private matching and set intersection. In C. Cachin and J. Camenisch, editors, *Advances in Cryptology - Eurocrypt '04*, volume 3024 of *LNCS*, pages 1–9. Springer-Verlag Berlin Heidelberg, 2004.

[8] O. Goldreich. *The Foundations of Cryptography*, volume 2. Cambridge University Press, 2004.

[9] S. Hohenberger and S. A. Weis. Honest-verifier private disjointness testing without random oracles. In *6th Workshop on Privacy Enhancing Technologies (PET'06)*, volume 4258 of *LNCS*, pages 277–294. Springer-Verlag Berlin Heidelberg, 2006.

[10] A. Kiayias and A. Mitrofanova. Testing disjointness and private datasets. In A. S. Patrick and M. Yung, editors, *Finanicial Cryptography (FC'05)*, volume 3570, pages 109–124. Springer-Verlag Berlin Heidelberg, 2005.

[11] A. Kiayias and A. Mitrofanova. Syntax-driven private evaluation of quantified membership queries. In *4th International Conference on Applied Cryptography and Network Security (ACNS'06)*, volume 3989 of *LNCS*, pages 470–485. Springer-Verlag, 2006.

[12] L. Kissner and D. Song. Privacy-preserving set operaitons. In V. Shoup, editor, *Advances in Cryptology - Crypto '05*, volume 3621 of *LNCS*, pages 241–257. Springer-Verlag Berlin Heidelberg, 2005.

[13] S. Laur, H. Lipmaa, and T. Mielikainen. Private itemset support counting. In *7th International Conference on Information and Communications Security*, volume 3783 of *LNCS*, pages 97–111. Springer-Verlag, 2005.

[14] H. Lipmaa. Verifiable homomorphic oblivious transfer and private equality test. In C. S. Laih, editor, *Advances in Cryptology - Asiacrypt '03*, volume 2894, pages 416–433. Springer-Verlag Berlin Heidelberg, 2003.

[15] P. Mohassel and M. Franklin. Efficient polynomial operations in the shared-coefficients setting. In M. Yung, editor, *Public Key Cryptography (PKC'06)*, volume 3958 of *LNCS*, pages 44–57. Springer-Verlag, 2006.

[16] M. Naor and B. Pinkas. Oblivious transfer and polynomial evaluation. In *31st annual ACM Symposium on Theory of Computing (STOC '99)*, pages 245–254, Atlanta, Georgia, May 1999.

[17] M. Naor and B. Pinkas. Distributed oblivious transfer. In T. Okanoto, editor, *Advances in Cryptology - Asiacrypt '00*, volume 1976 of *LNCS*, pages 205–219. Springer-Verlag, 2000.

[18] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology - Eurocrypt '99*, volume 1592 of *LNCS*, pages 223–238. Springer-Verlag, 1999.
[19] J. Pieprzyk, T. Hardjono, and J. Seberry. *Fundamentals of Computer Security*. Springer, 2003.
[20] Y. Sang, H. Shen, Y. Tan, and N. Xiong. Efficient protocols for privacy preserving matching against distributed datasets. In *8th International Conference of Information and Communications Security (ICICS'06)*, volume 4307 of *LNCS*, pages 210–227. Springer - Verlag, 2006.
[21] A. Shamir. How to share a secret. *Communications of the ACM*, 22:612–613, 1979.
[22] Y. Tsiounis and M. Yung. On the security of elgamal based encryption. In *Public Key Cryptography (PKC'98)*, volume 1431 of *LNCS*, pages 117–134. Springer-Verlag, 1998.
[23] A. C. Yao. Protocols for secure computations. In *23rd Symposium on Foundations of Computer Science (FOCS)*, pages 160–164. IEEE, 1982.

A Proof of Lemma 1

We have t participants and each of them has a share. Corresponding to the t points, the Vandermonde matrix V is constructed as follows:

$$V = \begin{pmatrix} 1 & x_{i_1} & \dots & x_{i_1}^{t-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{i_t} & \dots & x_{i_t}^{t-1} \end{pmatrix}.$$

Since the points are pairwise distinct, V is invertible. Let $V^{-1} = (v_{i,j})_{1 \le i,j \le t}$. By taking first row of V^{-1} and first column of V, we obtain $\sum_{j=1}^{t} v_{1,j} = 1$ as $V^{-1} \times V = \mathrm{Id}_t$ where Id_t denotes the $t \times t$ identity matrix.

The t polynomials $P_1(x), \dots, P_t(x)$ are defined as $P_j(x) := \prod_{\substack{1 \le k \le t \\ k \ne j}} \frac{x - x_{i_k}}{x_{i_k} - x_{i_j}}$ for any $1 \le j \le t$. Note that these polynomials have a nice property, namely

$$\forall j \in [1, \dots, t], \;\; P_j(x_{i_e}) = \begin{cases} 1 & \text{if } j = e \\ 0 & \text{otherwise} . \end{cases}$$

Those polynomials also can be rewritten as: $\forall j \in [1, \dots, t] \;\; P_j(x) = \sum_{k=1}^{t} a_{j,k}\, x^{k-1}$ where each $a_{j,k} \in \mathbb{Z}_{\mathrm{p}}$.

We now build a $t \times t$ matrix:

$$D = \begin{pmatrix} a_{1,1} & a_{2,1} & \cdots & a_{t,1} \\ \vdots & \vdots & \ddots & \vdots \\ a_{1,t} & a_{2,t} & \cdots & a_{t,t} \end{pmatrix}.$$

The j^{th} column of D represents the coefficients of $P_j(x)$. We claim that: $V^{-1} = D$. It is sufficient to prove that $V \times D$ is a identity matrix.

Let $V \times D = \mathcal{W} = (w_{\varsigma,\eta})_{\substack{1 \leq \varsigma \leq t \\ 1 \leq \eta \leq t}}$. We fix $\varsigma, \eta \in [1, \ldots, t]$ the coefficient $w_{\varsigma,\eta}$ is obtained by using the ς^{th} row of V along with the η^{th} column of D as $w_{\varsigma,\eta} = \sum_{m=1}^{t} x_{i_\varsigma}^{m-1} a_{\eta,m}$. Notice that $w_{\varsigma,\eta} = P_\eta(x_{i_\varsigma})$. Using the previous property of the polynomial, we obtain

$$w_{\varsigma,\eta} = \begin{cases} 1 & \text{if } \eta = \varsigma \\ 0 & \text{otherwise .} \end{cases}$$

This property demonstrates that $\mathcal{W}$ is a identity matrix, which proves that $V^{-1} = D$ as the inverse is unique.

Since the sum of the coefficients of the first row of V^{-1} is 1, we get $\sum_{j=1}^{t} v_{1,j} = \sum_{j=1}^{t} a_{j,1} = 1$. Notice that $a_{j,1}$ is the constant coefficient of $P_j(x)$, so $\forall j \in [1, \ldots, t],\ a_{j,1} = P_j(0) = \prod_{\substack{1 \leq k \leq t \\ k \neq j}} \frac{x_{i_k}}{x_{i_k} - x_{i_j}}$. Combining the previous two findings, we can conclude that:

$$\sum_{j=1}^{t} \left(\prod_{\substack{1 \leq k \leq t \\ k \neq j}} \frac{x_{i_k}}{x_{i_k} - x_{i_j}} \right) = 1.$$

B Proof of Theorem 1

Denote $\langle g \rangle$, the subgroup of $\mathbb{Z}_{\mathrm{p}}^*$ generated by g. By definition, the order of $\langle g \rangle$ is q.

The client $\mathcal{C}$ sends the group of t servers the encrypted values $E_{\text{pk}}(c_0), \ldots, E_{\text{pk}}(c_{m-1})$ where:

$$\forall i \in \{0, \ldots, m-1\} \quad E_{\text{pk}}(c_i) = (g^r, h^r\, f^{-c_i})$$

Thus, the group of t servers obtains:

$$g^r, h^r\, f^{-c_0}, \ldots, h^r\, f^{-c_{m-1}}$$

The elements g and h are two generators of the multiplicative group $\langle g \rangle$. As r is chosen uniformly at random over $\langle g \rangle$, g^r and h^r are two elements uniformly distributed over $\langle g \rangle$.

As the c_i's are all distinct, we get: $f^{-c_i} \not\equiv f^{-c_j} \bmod p$ when $i \neq j$. If $h^r \not\equiv 1 \bmod p$ then each element $h^r\, f^{-c_i} \bmod p$ is uniformly distributed over $\langle g \rangle$ and we have:

$$\Pr(h^r \not\equiv 1 \bmod p) = \Pr(r \not\equiv 0 \bmod q) = 1 - \frac{1}{q}$$

So, we deduce that $h^r\, f^{-c_0}, \ldots, h^r\, f^{-c_{m-1}}$ are m pairwise distinct elements uniformly distributed over $\langle g \rangle$ with probability $1 - \frac{1}{q}$ as the same value r is used for each of these elements.

As the discrete logarithm problem is assumed to be hard over $\mathbb{Z}_\mathrm{p}$ (DDH assumption), the group of t servers cannot compute r from g^r with non-negligible probability in polynomial time as a function of the bit size of p. Therefore, given the above analysis, we deduce that the t servers cannot distinguish the m elements $h^r\, f^{-c_0}, \ldots, h^r\, f^{-c_{m-1}}$ from m distinct elements of $\langle g \rangle$ drawn uniformly.

C Proof of Theorem 2

We first consider that $\mathcal{C}$ contacts t servers. At the end of Step 3.b, we have:

$$\forall \upsilon \in \{0, \ldots, m-1\} \quad \forall i \in \{0, \ldots, n-1\} \quad d_{\upsilon,i} = f^{\lambda \sum_{j=1}^{t} b_j\, (\mu_{i,\ell_j} - c_\upsilon)}$$

Using the proof of Lemma 2, we get:

$$\forall \upsilon \in \{0, \ldots, m-1\} \quad \forall i \in \{0, \ldots, n-1\} \quad d_{\upsilon,i} = f^{\lambda \left(\left(\sum_{j=1}^{t} b_j\, \mu_{i,\ell_j} \right) - c_\upsilon \right)}$$

Using that lemma, we deduce that, for each c_υ from $\mathcal{D}_\mathcal{C}$, we have:

$$c_\upsilon \in \mathcal{D}_\mathcal{P} \Longleftrightarrow \exists i_0 \in \{0, \ldots, n-1\} \quad \left(\sum_{j=1}^{t} b_j\, \mu_{i_0,\ell_j} \right) - c_\upsilon = 0$$

Now, assume that c_υ is not an element of $\mathcal{D}_\mathcal{P}$. We have:

$$\forall i \in \{0, \ldots, n-1\} \quad \left(\sum_{j=1}^{t} b_j\, \mu_{i,\ell_j} \right) - c_\upsilon \not\equiv 0 \bmod q$$

Since λ has been chosen uniformly at random from $\mathbb{Z}_\mathrm{q} - \{0\}$, we deduce that the element $\lambda \left(\left(\sum_{j=1}^{t} b_j\, \mu_{i,\ell_j} \right) - c_\upsilon \right)$ is uniformly distributed over $\mathbb{Z}_\mathrm{q} - \{0\}$ as well. As the discrete logarithm problem is assumed to be hard over $\mathbb{Z}_\mathrm{p}$ (DDH assumption), this exponent is not computable in polynomial time with non-negligible probability by $\mathcal{C}$ and thus the coefficients $d_{\upsilon,0}, \ldots, d_{\upsilon,n-1}$ appeared to be uniformly drawn from $\langle g \rangle$ to the client $\mathcal{C}$ as f generates that multiplicative group.

We now assume that $\mathcal{C}$ only contacted $t-1$ servers $S_{\ell_1}, \ldots, S_{\ell_{t-1}}$. In this situation, the polynomial $F(y)$ representing the provider dataset $\mathcal{D}_{\mathcal{P}}$ cannot be reconstructed uniquely identically to the secret polynomial of a (t, w)-Shamir secret sharing scheme when only $t-1$ participants work together. As a consequence, the missing participant involves that $F(y)$ can take p equally probable values where a single one is correct. Thus, $\mathcal{C}$ cannot recover $\mathcal{D}_{\mathcal{C}} \cap \mathcal{D}_{\mathcal{P}}$ even if he colludes with $t-1$ servers as he cannot reconstruct $F(y)$ and use (2) at Step 3.b.