

# Privacy-Preserving Distributed Set Intersection\*

Qingsong Ye<sup>1</sup>, Huaxiong Wang<sup>1,2</sup> and Christophe Tartary<sup>2,3</sup>

<sup>1</sup>Centre for Advanced Computing, Algorithms and Cryptography  
Department of Computing  
Macquarie University  
NSW 2109 Australia

<sup>2</sup>Division of Mathematical Sciences  
School of Physical and Mathematical Sciences  
Nanyang Technological University  
Singapore

<sup>3</sup>Institute for Theoretical Computer Science  
Tsinghua University  
Beijing, 100084  
People's Republic of China

qingsong@ics.mq.edu.au  
{HXWang, ctartary}@ntu.edu.sg

## 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 contacted server, the cardinality of set intersection can be computed efficiently. All protocols constructed in this paper are provably secure against an honest-but-curious adversary under the Decisional Diffie-Hellman assumption.

**Keywords:** Privacy-preserving Set Intersection, Homomorphic Encryption.

## 1 Introduction

Privacy-preserving set intersection protocols [7] are cryptographic techniques allowing two or more parties, each holding a set of inputs, to jointly calculate set intersection 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 a product  $P_1$  from  $C_1$  is also buying a 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 privacy-preserving set intersection 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 privacy-preserving set intersection in a distributed environment. We call this *privacy-preserving distributed set intersection*. 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 be able to find out the original data. Now, assume that a client holding her private dataset, wishes to compute the set intersection of the two

---

\*The original version of this paper appears in the proceedings of the 2nd International Workshop on Advances in Information Security (WAIS 2008) which are included into the proceedings of the 3rd International Conference on Availability, Security and Reliability (ARES 2008), p. 1332-1339, IEEE Computer Society.

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 intersection.

In general, privacy-preserving set intersection can be implemented using secure multi-party computation protocols [3, 24]. However, such solutions are generally inefficient. More specialized protocols on privacy-preserving set intersection are needed to improve its efficiency.

## 1.1 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 and the technique of oblivious polynomial evaluation [17]. To briefly describe the FNP construction, suppose  $CS = (K, E_{pk}, D_{sk})$  is a semantically secure public-key homomorphic encryption scheme. Assume that Alice has the dataset  $A = \{a_1, \dots, a_n\}$  and Bob owns the dataset  $B = \{b_1, \dots, b_m\}$ .

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  $E_{pk}(\alpha_i)$  using a homomorphic cryptosystem  $CS$  such as Paillier's [19] or the standard variant of the ElGamal encryption scheme (see [4]). Note that an homomorphic cryptosystem allows a party knowing  $E_{pk}(x)$  and  $E_{pk}(y)$  to compute  $E_{pk}(x + y) = E_{pk}(x) \cdot E_{pk}(y)$  and  $E_{pk}(x \cdot c) = E_{pk}(x)^c$  where  $c$  is any constant. The reader is referred to Section 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 obviously evaluate  $E_{pk}(f(b_i))$  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  $E_{pk}(f(b_i))^r = E_{pk}(r \cdot f(b_i))$ . Consequently, if  $f(b_i) = 0$  then the encryption of  $E_{pk}(r \cdot f(b_i)) = E_{pk}(0)$ . Otherwise,  $E_{pk}(r \cdot f(b_i))$  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  $E_{pk}(r \cdot f(b_i) + b_i)$ 's to her. She decrypts them and tests whether any of the resulting plaintexts are in  $A$  as  $D_{sk}(E_{pk}(r \cdot f(b_i) + b_i)) = b_i$  if and only if  $b_i \in A$ .

Inspired by FNP, Kissner and Song [12] propose a solution to various privacy-preserving set operations such as set union, set intersection, 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 [21].

Protocols for testing the subset relation in a two-party setting are discussed in [11, 14] 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, 17, 15].

## 1.2 Our Results

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

Contrary to the previous two-party privacy-preserving set intersection 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 a single server is a great concern.

We first compute the  $w$  shares of the dataset  $B$  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 [22] a single time to compute the shares of the whole dataset.

In our set intersection protocol, we will use our observation that  $\sum_{j=1}^t c_j (b - b_{\ell_j}) = 0$  where the  $c_j$ 's are Lagrange interpolation coefficients,  $b$  is a value and the  $b_{\ell_j}$ 's are the shares of  $b$ . Using this relation, one can obviously check that an element  $b'$  is equal to  $b$  by collecting  $t$  values  $r(b' - b_{\ell_j})$  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  $A$ , she is able to compute  $A \cap B$  privately.

We then extend our privacy-preserving distributed set intersection solution to a one-round protocol evaluating  $|A \cap B|$  only. To prevent the client from learning the intersection  $A \cap B$ , each server will permute the cryptograms before sending

them back to the client  $\mathcal{C}$ . Thus, after decryption and computation, the client only learns  $|A \cap B|$ .

Our protocols are secure against an honest-but-curious 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 [23]. 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  $O(t|A||B| \times \log_2 p)$  bits. The computation cost complexity is  $O(t|A||B| \times \log_2^3 p)$  bits for our two protocols. These complexity results are efficient considering our distributed setting.

Our paper is organized as follows. In Section 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 Section 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 Section 4, we give concluding remarks.

## 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  $K$  and security parameter  $\chi$ , encryption algorithm  $E_{pk}$  and decryption algorithm  $D_{sk}$  is said to be **homomorphic** if for every key pair  $(pk, sk) \in K(1^\chi)$ :

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 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  $e$  of the plaintext  $m$  is transformed into  $e'$  such that  $e'$  is still a valid cryptogram for the message  $m$  but relying on a different random string from  $e$ 's.

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, 15]. In our protocols, the computations are carried out over  $\mathbb{Z}_p$  where  $p$  is prime. We assume that  $p = 2q + 1$  where  $q$  is also prime.

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

$$\begin{aligned} E_{pk}(r_1, m_1) \odot E_{pk}(r_2, m_2) &:= (g^{r_1+r_2}, h^{r_1+r_2} f^{m_1+m_2}) \\ &= E_{pk}(r_1 + r_2, m_1 + m_2) \end{aligned}$$

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

$$\begin{aligned} E_{pk}(r, m)^c &= \underbrace{E_{pk}(r, m) \odot E_{pk}(r, m) \odot \dots \odot E_{pk}(r, m)}_{c \text{ times}} \\ &:= E_{pk}(cr, cm) \end{aligned}$$

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

## 2.2 Distributed Environment

The players are a client  $\mathcal{C}$ , a provider  $\mathcal{P}$ , and  $w$  servers  $S_1, S_2, \dots, S_w$ . We assume that the provider holds a dataset  $B = \{b_0, b_1, \dots, b_{n-1}\}$  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 operation of the set intersection. Instead,  $\mathcal{C}$  contacts at least  $t$  servers to discover the set intersection. Note that this distributed setting was first proposed by Naor and Pinkas [18].

Our homomorphic encryption system is based on a variant of the ElGamal cryptosystem where the message space is over  $\mathbb{Z}_q$  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  $B$ , i.e.:

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

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

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

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  $Q(\ell, y) = \sum_{i=0}^{n-1} \beta_{i,\ell} y^i$  to server  $S_\ell$  where  $\forall i \in \{0, \dots, n-1\} \ \beta_{i,\ell} = b_i + \vartheta_{i,\ell}$  with  $\vartheta_{i,\ell} = \sum_{j=1}^{t-1} \alpha_{j,i} \ell^j$ . The server  $S_\ell$  receives a set of shared coefficients  $\{\beta_{0,\ell}, \dots, \beta_{n-1,\ell}\}$  of the polynomial  $F(y)$  (see [16]).

**Secret Reconstruction Phase.** We now show how any  $t$ -subset of servers can recover  $F(y)$ . Denote  $S_{\ell_1}, \dots, S_{\ell_t}$  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  $Q(\ell_j, y)$  for  $j \in \{1, \dots, t\}$  verify the following system:

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

where  $V$  is the  $t \times t$  Vandermonde matrix [13] defined as:  $V := (\ell_i^j)_{i=1, \dots, t}^{j=0, \dots, t-1}$ . Since we are only interested in the reconstruction of  $F(y)$ , we simply need to know the first row of  $V^{-1}$ ,  $(v_{1,1} \cdots v_{1,t})$ . Then, we have:

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

As a consequence, we obtain:

$$\forall i \in \{0, \dots, n-1\} \quad \sum_{j=1}^t v_{1,j} \beta_{i,\ell_j} = b_i \quad (1)$$

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

**Lemma 1** *We have:*

$$\forall j \in \{1, \dots, t\} \quad v_{1,j} = \prod_{\substack{1 \leq k \leq t \\ k \neq j}} \frac{\ell_k}{\ell_k - \ell_j} \quad \text{and} \quad \sum_{j=1}^t v_{1,j} = 1$$

From the previous lemma, we deduce:

$$\forall i \in \{1, \dots, n-1\} \quad b_i = \sum_{j=1}^t \left( \prod_{\substack{1 \leq k \leq t \\ k \neq j}} \frac{\ell_k}{\ell_k - \ell_j} \right) \beta_{i,\ell_j}$$

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  $\mathbb{Z}_p$ . 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 a different modulus, the congruence will be written in full to avoid confusion.

## 2.3 Adversary Model

We consider an honest-but-curious 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 ( $t$ -security)** A distributed protocol is said to be  $t$ -secure if among  $w$  servers any subset of  $t - 1$  corrupted servers learn no information about the provider's dataset and the protocol result.

Following [17, 18] our model should meet the following three requirements:

1. **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.
2. **Client's security.** A 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.
3. **Provider's security.** A 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 Privacy-Preserving Distributed Set Intersection

In this section, we address the problem of designing protocols for privacy-preserving distributed set intersection related issues. Those targeted in this paper are the privacy preserving set intersection problem and the cardinality of set intersection problem. Let the dataset  $A$  of the client  $\mathcal{C}$  be  $\{a_0, \dots, a_{m-1}\}$ . The provider  $\mathcal{P}$  broadcasts  $\lambda \xleftarrow{R} \mathbb{Z}_q - \{0\}$  to the  $w$  servers  $S_1, \dots, S_w$ , and also distributes the shares of his dataset  $B = \{b_0, \dots, b_{n-1}\}$  to  $w$  servers as in Section 2.2. Note that it is assumed that  $|A| = m$  and  $|B| = n$  are publicly known.

### 3.1 Determination of the Set Intersection

Algorithm 1 represents a protocol which enables the client  $\mathcal{C}$  to compute the intersection  $A \cap B$  by contacting any subset of  $t$  servers  $S_{\ell_1}, \dots, S_{\ell_t}$ .

---

#### Algorithm 1 Privacy-Preserving Set Intersection

---

**Input:** The client  $\mathcal{C}$  has a set of data  $A$ . Each server  $S_\ell$  ( $1 \leq \ell \leq w$ ) knows the random value  $\lambda$  and the shared coefficients  $\{\beta_{0,\ell}, \dots, \beta_{n-1,\ell}\}$  of the polynomial  $F(y)$  (whose coefficients are the elements of the provider's dataset  $B$ ).

1.  $\mathcal{C}$  generates a new key pair  $(pk, sk) \leftarrow K(1^\times)$ , and then broadcasts  $\{E_{pk}(a_0), \dots, E_{pk}(a_{m-1})\}$  with her public key to  $t$  servers  $S_{\ell_1}, \dots, S_{\ell_t}$ .
2. For  $j = 1, \dots, t$ , each contacted server  $S_{\ell_j}$  computes and sends  $E_{pk}(\lambda(a_\iota - \beta_{i,\ell_j}))$  to  $\mathcal{C}$  for  $\iota \in \{0, \dots, m-1\}$  and  $i \in \{0, \dots, n-1\}$ .
3. For  $\iota = 0, \dots, m-1$ , the client  $\mathcal{C}$  does the following:
  - 3.1. He computes  $d_{\iota,i,j} \leftarrow D_{sk}(E_{pk}(\lambda(a_\iota - \beta_{i,\ell_j})))$  for  $i \in \{0, \dots, n-1\}$  and  $j \in \{1, \dots, t\}$ .
  - 3.2. He computes  $d_{\iota,i} \leftarrow \prod_{j=1}^t (d_{\iota,i,j})^{c_j}$  for  $i = 0, \dots, n-1$ , where  $c_j$ 's are the Lagrange interpolation coefficients.
  - 3.3. He concludes  $a_\iota \in B$ , if  $d_{\iota,i} = 1$  for  $i \in [0, \dots, n-1]$ ; otherwise  $d_{\iota,i}$  is a random integer.

**Output:** The client  $\mathcal{C}$  learns  $A \cap B$ .

---

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

**Lemma 2** Let  $c_j = \prod_{\substack{1 \leq k \leq t \\ k \neq j}} \frac{\ell_k}{\ell_k - \ell_j}$  be the Lagrange interpolation coefficient in Algorithm 1. Then:

$$\sum_{j=1}^t c_j (b_i - \beta_{i, \ell_j}) = 0$$

*Proof.*

Note that the coefficient  $c_j$  corresponds to the  $j^{\text{th}}$  coefficient of the first row of the matrix  $V^{-1}$  denoted  $v_{1,j}$  in Section 2.2.

From Lemma 1, we get that  $\sum_{j=1}^t c_j = 1$ . Thus, Equation (1) provides our result. □

In the above protocol, the client  $\mathcal{C}$  first encrypts each element  $a_\iota$  of her dataset by using her public key as  $E_{\text{pk}}(a_\iota)$  for  $\iota \in \{0, \dots, m-1\}$  and broadcasts all these encrypted elements to  $t$  servers. For each encrypted element  $E_{\text{pk}}(a_\iota)$ , the servers  $S_{\ell_j}$  ( $1 \leq j \leq t$ ) compute  $E_{\text{pk}}(\lambda(a_\iota - \beta_{i, \ell_j}))$  for  $i \in \{0, \dots, n-1\}$ , and send all the  $E_{\text{pk}}(\lambda(a_\iota - \beta_{i, \ell_j}))$ 's back to

$\mathcal{C}$ . The client  $\mathcal{C}$  then decrypts those  $E_{\text{pk}}(\lambda(a_\iota - \beta_{i, \ell_j}))$ 's and computes  $d_{\iota, i} = f^{\lambda \sum_{j=1}^t c_j (a_\iota - \beta_{i, \ell_j})}$  for each  $i = 0, \dots, n-1$ .

Note that if  $a_\iota = b_i$  then  $\sum_{j=1}^t c_j (a_\iota - \beta_{i, \ell_j}) = 0$ . Therefore, the client  $\mathcal{C}$  learns that  $a_\iota \in B$  if there exists  $i \in \{0, \dots, n-1\}$

such that  $f^{\lambda \sum_{j=1}^t c_j (a_\iota - \beta_{i, \ell_j})} = 1$ . When all the steps are finished,  $\mathcal{C}$  learns  $A \cap B$ .

**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 Appendix C.

**Theorem 1** Given the set intersection protocol described in Algorithm 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  $\{\beta_{0, \ell}, \dots, \beta_{n-1, \ell}\}$  ( $1 \leq \ell \leq w$ ) 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 privacy-preserving distributed set intersection protocol, we develop an algorithm only computing the cardinality of the datasets' intersection  $|A \cap B|$ . Denote  $\mathbb{P}_{m n}$  the set of all permutations of  $\{1, \dots, m n\}$ . Assume that  $\mathcal{P}$  has a private permutation function  $\pi$ , chosen uniformly at random from  $\mathbb{P}_{m n}$ , which is given to the  $w$  servers. This scheme is represented as Algorithm 2.

This protocol works in the same way as the distributed set intersection protocol with the addition that all servers run the same permutation function  $\pi$  on their computed cryptograms. This is to prevent the client  $\mathcal{C}$  from learning the set intersection  $A \cap B$ .

**Security of the Construction.** The security model and the proof of this protocol are similar to our set-intersection protocol presented in Section 3.1 as the permutation  $\pi$  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  $S_{\ell_j}$  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  $\mathbb{Z}_p$  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.

---

**Algorithm 2** Privacy-Preserving cardinality of Set Intersection

---

**Input:** The client  $\mathcal{C}$  has a set of data  $A$ . Each server  $S_\ell$  ( $1 \leq \ell \leq w$ ) knows the random value  $\lambda$ , the permutation function  $\pi$  and the shared coefficients  $\{\beta_{0,\ell}, \dots, \beta_{n-1,\ell}\}$  of the polynomial  $F(y)$  (whose coefficients are the elements of the provider's dataset  $B$ ).

1.  $\mathcal{C}$  generates a new key pair  $(pk, sk) \leftarrow K(1^\lambda)$ , and then broadcasts  $\{E_{pk}(a_0), \dots, E_{pk}(a_{m-1})\}$  with her public key to  $t$  servers  $S_{\ell_1}, \dots, S_{\ell_t}$ .
2. For  $j = 1, \dots, t$  each contacted server  $S_{\ell_j}$  does the following:
  - 2.1. He computes  $\tau_{\nu,i,j} \leftarrow E_{pk}(\lambda(a_\nu - \beta_{i,\ell_j}))$  for  $\nu \in \{0, \dots, m-1\}$  and  $i \in \{0, \dots, n-1\}$ .
  - 2.2. He gets  $\{\tau_{\pi(0,0),j}, \dots, \tau_{\pi(0,n-1),j}, \tau_{\pi(1,0),j}, \dots, \tau_{\pi(m-1,n-1),j}\} \leftarrow \pi(\tau_{0,0,j}, \dots, \tau_{0,n-1,j}, \tau_{1,0,j}, \dots, \tau_{m-1,n-1,j})$ .
  - 2.3. He sends  $\{\tau_{\pi(0,0),j}, \dots, \tau_{\pi(0,n-1),j}, \tau_{\pi(1,0),j}, \dots, \tau_{\pi(m-1,n-1),j}\}$  to  $\mathcal{C}$ .
3. For  $\nu' = 0, \dots, m-1$ , the client  $\mathcal{C}$  does the following:
  - 3.1. He computes  $d_{\pi(\nu',i),j} \leftarrow D_{sk}(\tau_{\pi(\nu',i),j})$  for  $i \in \{0, \dots, n-1\}$  and  $j \in \{1, \dots, t\}$ .
  - 3.2. He computes  $d_{\pi(\nu',i)} \leftarrow \prod_{j=1}^t (d_{\pi(\nu',i),j})^{c_j}$  for  $i \in \{0, \dots, n-1\}$ , where  $c_j$ 's are Lagrange interpolation coefficients.
4. When this process concludes,  $\mathcal{C}$  learns  $|A \cap B|$  as it is the number of  $d_{\pi(\nu',i)}$ 's equal to 1.

**Output:** The client  $\mathcal{C}$  learns  $|A \cap B|$ .

---

Note that each decryption represents one modular multiplication and one modular exponentiation. Each server  $S_{\ell_j}$  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  $\lceil \log_2(p-1) \rceil$  modular multiplications using the Fast Exponentiation algorithm presented in [20].

The cost of our second protocol is the cost of the first one plus  $t$  executions of the permutation  $\pi$ . Assuming that  $\pi$  is represented by its binary permutation matrix  $M_\pi$ , each of these  $t$  queries has a negligible cost since  $\pi$  is a simple reordering of its inputs ( $M_\pi$  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. 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, 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.

## Acknowledgment

The authors are grateful to the anonymous reviewers for their comments to improve the quality of this paper. This work was supported by the Australian Research Council under ARC Discovery Projects DP0558773, DP0665035 and DP0663452. This work was supported in part by the National Natural Science Foundation of China Grant 60553001 and the National Basic Research Program of China Grant 2007CB807900, 2007CB807901. Qingsong Ye's work was funded by a iMURS scholarship provided by Macquarie University. Christophe Tartary did most of his work while at Macquarie University where his research was also supported by an iMURS scholarship. The research of Huaxiong Wang is partially supported by the Ministry of Education of Singapore under grant T206B2204.

## References

- [1] Ben Adida and Douglas Wikstrom. How to shuffle in public. In *4th Theory of Cryptography Conference*, volume 4392 of *Lecture Notes in Computer Science*, pages 555 – 574, Amsterdam, The Netherlands, February 2007. Springer-Verlag.
- [2] Bill Aiello, Yuval Ishai, and Omer Reingold. Priced oblivious transfer: How to sell digital goods. In *Advances in Cryptology - Eurocrypt'01*, volume 2045 of *Lecture Notes in Computer Science*, pages 119 – 135, Aarhus, Denmark, May 2001. Springer-Verlag.
- [3] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *20th Annual ACM Symposium on Theory of Computing*, pages 1 – 10, Chicago, USA, May 1988. ACM Press.
- [4] Ronald Cramer, Rosario Gennaro, and Berry Schoenmakers. A secure and optimally efficient multi-authority election scheme. In *Advances in Cryptology - Eurocrypt'97*, volume 1233 of *Lecture Notes in Computer Sciences*, pages 103 – 118, Konstanz, Germany, May 1997. Springer - Verlag.
- [5] Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In G. R. Blakley and David Chaum, editors, *Advances in Cryptology - Crypto'84*, volume 196 of *Lecture Notes in Computer Science*, pages 19 – 22, Santa Barbara, USA, August 1984. Springer-Verlag.
- [6] Ronald Fagin, Moni Naor, and Peter Winkler. Comparing information without leaking it. *Communications of the ACM*, 39(5):77 – 85, May 1996.
- [7] Michael J. Freedman, Kobbi Nissim, and Benny Pinkas. Efficient private matching and set intersection. In *Advances in Cryptology - Eurocrypt'04*, volume 3024 of *Lecture Notes in Computer science*, pages 1 – 9, Interlaken, Switzerland, May 2004. Springer-Verlag.
- [8] Oded Goldreich. *Foundations of Cryptography: Volume II - Basic Applications*. Cambridge University Press, 2004.
- [9] Susan Hohenberger and Stephen A. Weis. Honest-verifier private disjointness testing without random oracles. In *6th Workshop on Privacy Enhancing Technologies*, volume 4258 of *Lecture Notes in Computer Science*, pages 277 – 294, Cambridge, UK, June 2006. Springer-Verlag.
- [10] Aggelos Kiayias and Antonina Mitrofanova. Testing disjointness and private datasets. In *9th International Conference on Financial Cryptography and Data Security*, volume 3570 of *Lecture Notes in Computer Science*, pages 109 – 124, Roseau, The Commonwealth Of Dominica, March 2005. Springer-Verlag.
- [11] Aggelos Kiayias and Antonina Mitrofanova. Syntax-driven private evaluation of quantified membership queries. In *4th International Conference on Applied Cryptography and Network Security*, volume 3989 of *Lecture Notes in Computer Science*, pages 470 – 485, Singapore, June 2006. Springer-Verlag.
- [12] Lea Kissner and Dawn Song. Privacy-preserving set operations. In *Advances in Cryptology - Crypto'05*, volume 3621 of *Lecture Notes in Computer Science*, pages 241 – 257, Santa Barbara, USA, August 2005. Springer-Verlag.
- [13] Donald E. Knuth. *The Art of Computer Programming: Volume I - Fundamental Algorithms (Third Edition)*. Addison - Wesley, 1997.
- [14] Sven Laur, Helger Lipmaa, and Taneli Mielikainen. Private itemset support counting. In *7th International Conference on Information and Communications Security*, volume 3783 of *Lecture Notes in Computer Science*, pages 97 – 111, Beijing, P. R. China, December 2005. Springer-Verlag.
- [15] Helger Lipmaa. Verifiable homomorphic oblivious transfer and private equality test. In *Advances in Cryptology - Asiacrypt'03*, volume 2894 of *Lecture Notes in Computer Science*, pages 416 – 433, Taipei, Taiwan, November 2003. Springer-Verlag.
- [16] Payman Mohassel and Matthew Franklin. Efficient polynomial operations in the shared-coefficients setting. In *Public Key Cryptography*, volume 3958 of *Lecture Notes in Computer Science*, pages 44 – 57, New York, USA, April 2006. Springer-Verlag.
- [17] Moni Naor and Benny Pinkas. Oblivious transfer and polynomial evaluation. In *31st annual ACM Symposium on Theory of Computing*, pages 245 – 254, Atlanta, USA, May 1999. ACM Press.
- [18] Moni Naor and Benny Pinkas. Distributed oblivious transfer. In *Advances in Cryptology - Asiacrypt '00*, volume 1976 of *Lecture Notes in Computer Science*, pages 205 – 219, Kyoto, Japan, December 2000. Springer-Verlag.

- [19] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology - Eurocrypt'99*, volume 1592 of *Lecture Notes in Computer Science*, pages 223 – 238, Prague, Czech Republic, May 1999. Springer-Verlag.
- [20] Josef Pieprzyk, Thomas Hardjono, and Jennifer Seberry. *Fundamentals of Computer Security*. Springer, 2003.
- [21] Yingpeng Sang, Hong Shen, Yasuo Tan, and Naixue Xiong. Efficient protocols for privacy preserving matching against distributed datasets. In *8th International Conference of Information and Communications Security*, volume 4307 of *Lecture Notes in Computer Science*, pages 210 – 227, Raleigh, USA, December 2006. Springer - Verlag.
- [22] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612 – 613, November 1979.
- [23] Yiannis Tsiounis and Moti Yung. On the security of ElGamal based encryption. In *Public Key Cryptography*, volume 1431 of *Lecture Notes in Computer Science*, pages 117 – 134, Yokohama, Japan, February 1998. Springer-Verlag.
- [24] Andrew Chi-Chih Yao. Protocols for secure computations. In *23rd Annual IEEE Symposium on Foundations of Computer Science*, pages 80 – 91, Chicago, USA, November 1982. IEEE Press.

## A Proof of Lemma 1

Assume that  $t$  participants pool their shares together. The Vandermonde matrix  $V$  corresponding to these participants 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  $x_{i_j}$ 's are pairwise distinct,  $V$  is invertible. Let  $V^{-1} = (v_{i,j})_{1 \leq i,j \leq t}$ . By taking first row of  $V^{-1}$  and first column of  $V$ , we obtain  $\sum_{j=1}^t v_{1,j} = 1$  as  $V^{-1} \times V = \text{Id}_t$  where  $\text{Id}_t$  denotes the  $t \times t$  identity matrix.

Let  $P_1(x), \dots, P_t(x)$  be  $t$  polynomials, such that  $P_j(x) := \prod_{\substack{1 \leq k \leq t \\ k \neq j}} \frac{x - x_{i_k}}{x_{i_k} - x_{i_j}}$  for any  $1 \leq j \leq t$ . Note that these polynomials have a nice property, namely:

$$\forall j \in \{1, \dots, t\} \quad P_j(x_{i_{j'}}) = \begin{cases} 1 & \text{if } j = j' \\ 0 & \text{otherwise} \end{cases}$$

Those polynomials also can be rewritten as:  $\forall j \in \{1, \dots, t\} \quad P_j(x) = \sum_{k=1}^t \gamma_{j,k} x^{k-1}$  where each  $\gamma_{j,k} \in \mathbb{Z}_p$ . We now build a  $t \times t$  matrix:

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

The  $j^{\text{th}}$  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  $V \times D = \mathcal{W} = (\omega_{\varsigma,\eta})_{\substack{1 \leq \varsigma \leq t \\ 1 \leq \eta \leq t}}$ . Fixing  $\varsigma, \eta \in \{1, \dots, t\}$ , the coefficient  $\omega_{\varsigma,\eta}$  is obtained by using the  $\varsigma^{\text{th}}$  row of  $V$  along with the  $\eta^{\text{th}}$  column of  $D$  as  $\omega_{\varsigma,\eta} = \sum_{\rho=1}^t x_{i_\varsigma}^{\rho-1} \gamma_{\eta,\rho}$ . Notice that  $\omega_{\varsigma,\eta} = P_\eta(x_{i_\varsigma})$ . Using the previous property of the polynomial, we obtain:

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

This property demonstrates that  $\mathcal{W}$  is an 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:

$$\sum_{j=1}^t v_{1,j} = \sum_{j=1}^t \gamma_{j,1} = 1$$

Notice that  $\gamma_{j,1}$  is the constant coefficient of  $P_j(x)$ , so:

$$\forall j \in \{1, \dots, t\} \quad \gamma_{j,1} = P_j(0) = \prod_{\substack{1 \leq k \leq t \\ k \neq j}} \frac{x_{i_k}}{x_{i_k} - x_{i_j}}$$

Combining the previous two results, we deduce:

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

## B Proof of Theorem 1

Denote  $\langle g \rangle$ , the subgroup of  $\mathbb{Z}_p^*$  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  $E_{\text{pk}}(a_0), \dots, E_{\text{pk}}(a_{m-1})$  where:

$$\forall \iota \in \{0, \dots, m-1\} \quad E_{\text{pk}}(a_\iota) = (g^r, h^r f^{a_\iota})$$

Thus, the group of  $t$  servers obtains:

$$g^r, h^r f^{a_0}, \dots, h^r f^{a_{m-1}}$$

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  $a_\iota$ 's are all distinct, we get:  $f^{a_\iota} \not\equiv f^{a_{\iota'}} \pmod p$  when  $\iota \neq \iota'$ . If  $h^r \not\equiv 1 \pmod p$  then each element  $h^r f^{a_\iota} \pmod p$  is uniformly distributed over  $\langle g \rangle$  and we have:

$$\text{Prob}(h^r \not\equiv 1 \pmod p) = \text{Prob}(r \not\equiv 0 \pmod q) = 1 - \frac{1}{q}$$

So, we deduce that  $h^r f^{a_0}, \dots, h^r f^{a_{m-1}}$  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  $\mathbb{Z}_p$  (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  $h^r f^{a_0}, \dots, h^r f^{a_{m-1}}$  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 3.2, we have:

$$\forall \iota \in \{0, \dots, m-1\} \quad \forall i \in \{0, \dots, n-1\} \quad d_{\iota,i} = f^{\lambda \sum_{j=1}^t c_j (a_\iota - \beta_{i,\ell_j})}$$

Using the proof of Lemma 2, we get:

$$d_{\iota,i} = f^{\lambda \left( a_\iota - \left( \sum_{j=1}^t c_j \beta_{i,\ell_j} \right) \right)}$$

Using that lemma, we deduce that, for each  $a_\iota$  from  $A$ , we have:

$$a_\iota \in B \iff \exists i_0 \in \{0, \dots, n-1\} \quad a_\iota - \left( \sum_{j=1}^t c_j \beta_{i_0,\ell_j} \right) = 0$$

Now, assume that  $a_\iota$  is not an element of  $B$ . We have:

$$\forall i \in \{0, \dots, n-1\} \quad a_\iota - \left( \sum_{j=1}^t c_j \beta_{i,\ell_j} \right) \neq 0$$

Since  $\lambda$  has been chosen uniformly at random from  $\mathbb{Z}_q - \{0\}$ , we deduce that the element  $\lambda \left( a_\iota - \left( \sum_{j=1}^t c_j \beta_{i, \ell_j} \right) \right)$  is uniformly distributed over  $\mathbb{Z}_q - \{0\}$  as well. As the discrete logarithm problem is assumed to be hard over  $\mathbb{Z}_p$  (DDH assumption), this exponent is not computable in polynomial time with non-negligible probability by  $\mathcal{C}$  and thus the coefficients  $d_{\iota,0}, \dots, d_{\iota,n-1}$  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  $S_{\ell_1}, \dots, S_{\ell_{t-1}}$ . In this situation, the polynomial  $F(y)$  representing the provider dataset  $B$  cannot be reconstructed uniquely 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  $A \cap B$  even if he colludes with  $t - 1$  servers as he cannot reconstruct  $F(y)$  and use Equation (1) at Step 3.2.