Privacy-Preserving Distributed Set Operations

Qingsong Ye and Huaxiong Wang

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

Abstract. Motivated by the demand of databases outsourcing and its security concerns, we investigate privacy-preserving set operations in a distributed scenario. By combining Shamir secret sharing scheme and homomorphic encryption, we propose a one-round protocol for privacy-preserving distributed set intersection. We then show that, with an additional round of interaction, cardinality of distributed set-intersections can be computed efficiently. Moreover, by using oblivious polynomial evaluation techniques, we provide a solution for distributed subset relation problem. All protocols constructed in this paper are provably secure against a semi-honest adversary under the Decisional Diffie-Hellman assumption.

Keywords: privacy-preserving set-operation, homomorphic encryption

1 Introduction

Privacy-Preserving Set Operation [9] is a cryptographic technique, which allows two or more parties each holding a set of inputs to jointly calculate set operations of their inputs without leaking any information. Imagine that two companies want to analyze their customers consuming trends. That is, they want to determine the likelihood that a customer buying the product P_1 from the company C_1 is also buying the product P_2 from the company C_2. To gain this information, they would like to perform a set-intersection operation on two private sets. In order to preserve the companies business secrets and to protect the customers privacy, the customers details must not be revealed. Another example is when a research institute and a group of hospitals cooperate to analyze patients records anonymously. Currently, there is a number of secure set operation protocols available [4, 9, 8].

Motivated by the demand of databases outsourcing and security concerns on its applications, we investigate privacy-preserving set operations in a distributed scenario. We call this Privacy-Preserving Distributed Set Operations. To illustrate this problem consider following example. Suppose a *Provider* who owns a set of data wants to distribute his dataset to w servers by using a secret sharing scheme. A *Client* who possesses a different dataset wishes to compute certain set operations between these two datasets with the cooperation of a threshold of the servers. The additional requirement is that this must be done with a minimum possible disclosure of information, only using certain set operations which are discovered by the client at the end of the protocol. Considering the high-cost of database encryption and the complex process of querying such an encrypted database, the constructions we describe are useful in the case of database outsourcing where an individual server is not trusted by the provider.

Our Contribution. In this paper, we propose an efficient technique for privacy preserving set operations in the distributed setting. Our approach is based on homomorphic encryption and oblivious polynomial evaluation, which are effective and computationally secure. We design an efficient method to enable privacy-preserving computation of *distributed set intersection* based on Lagrange's interpolation. We then apply the oblivious polynomial evaluation technique to construct a protocol for testing whether a given set is a subset of a distributed set. The protocol is efficient and requires only one more round communication between the client and anyone of w servers. To avoid the computational cost of oblivious polynomial evaluation, we employ the technique of random shuffling and provide efficient solutions on *cardinality of distributed set-intersection* problem.

Our protocols are secure against a semi-honest adversary. Such an adversary follows the steps of the protocol but tries to learn extra information from the messages it receives during every round of the protocol. Our homomorphic encryption is based on the ElGamal scheme [5], which is semantically secure if the Decisional Diffie-Hellman (DDH) assumption holds [16]. Note that as in [4, 8] our protocols reveal the size of the datasets of both the client and the provider. As suggested in [8], "dummy" elements can be used for dataset padding in order to hide the size of the dataset.

Because of space limitation all the proofs in this paper are only sketched.

Related Work. In theory, the privacy-preserving set operation problems can be solved by secure Multi-Party Computation (MPC) [2, 17]. However, the solution from general secure MPC are typically not efficent. Kessner and Song [9] proposed a solution to various privacy-preserving set operation problems in the context of private computation on multisets for multi-players. Based on a threshold homomorphic cryptosystem, Sang et al. [14] improved the computation and communication complexity on the set intersection and set matching problems.

The problem of securely computing the intersection of two private datasets was considered in [12]. Recently, Freedman et. al. [4] proposed efficient protocols for the problem of private set-intersection, based on the representation of datasets as roots of a polynomial. Private disjointness test of two datasets are discussed in [8, 7]. Protocols for private equality tests are a special case of the private disjointness test, where each party has a single element in the database. These were considered in [3, 12, 10].

Our paper is organized as follows. In Section 2 we introduce the cryptographic primitives, distributed setting, and adversary model. In Section 3 we present a protocol and security analysis for the distributed set intersection problem. Section 4 gives a protocol and security analysis for the distributed subset relation problem. To extend our solutions on the distributed set intersection and the distributed subset relation problems, we provide solutions for the cardinality of distributed set-intersection problem in Section 5. Finally, in Section 6 we give concluding remarks and discuss possible future work.

2 Preliminaries

2.1 Additively Homomorphic Encryption

We will utilize an additively homomorphic public-key cryptosystem. Let $E_{\text{pk}}(\cdot)$ denote the encryption function with a public key pk. The cryptosystem supports the following operations, which can be performed without knowing the private key.

- Given $E_{\text{pk}}(a)$ and $E_{\text{pk}}(b)$, we can efficiently compute $E_{\text{pk}}(a+b)$.
- Given a constant c and $E_{\text{pk}}(a)$, we can efficiently compute $E_{\text{pk}}(ca)$.

In our schemes, the computations are carried out over $\mathbb{Z}_{\text{p}}$ where p is a prime. We note that all of our constructions can be based on the standard variant of ElGamal encryption. This variant has been employed recently for constructing the protocols of privacy-preserving set operations (see, for example [8, 1, 10]). Let the triple (K, E, D) be the variant of ElGamal where

- K is the key-generation algorithm. Given a security parameter $l = \log_2 p$, $(pk, sk) \leftarrow K(1^l)$ where the public-key $pk := \langle p, g, h, f \rangle$ and the corresponding secret-key $x = \log_{\text{g}} h$. More precisely, $q = (p-1)/2$ is also a prime number, g is an element of order q in $\mathbb{Z}_{\text{p}}^*$ and $h, f \in \langle g \rangle$;
- E is the encryption algorithm. Given the public-key pk and a plaintext, one encrypts a plaintext $m \in \mathbb{Z}_{\text{q}}$ as $E_{\text{pk}}(r, m) = (g^r, h^r f^m)$ where $r \overset{\text{R}}{\leftarrow} \mathbb{Z}_{\text{q}}$;
- D is the decryption algorithm. Given the secret-key x and a ciphertext (a, b) the decryption algorithm returns $a^{-x}b \mod p$. Notice that this will only return f^m rather than m, however this suffices for our setting. In our protocols we are only interested in testing whether $m = 0$, which is equivalent to testing if $a^{-x}b \equiv 1 \mod p$.

To demonstrate the above encryption scheme is indeed homomorphic, we define a operation $\odot$ as multiplication over $\mathbb{Z}_{\text{p}} \times \mathbb{Z}_{\text{p}}$. Let $m_1, m_2, m \in \mathbb{Z}_{\text{q}}$ and corresponding $r_1, r_2, r \overset{\text{R}}{\leftarrow} \mathbb{Z}_{\text{q}}$, then

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

If we repeat this operation c times, we have

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

For brevity, 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 $r \overset{\text{R}}{\leftarrow} \mathbb{Z}_{\text{q}}$.

2.2 Oblivious Polynomial Evaluation

There are two parties, $\mathcal{A}$ and $\mathcal{B}$. $\mathcal{A}$ owns a function $\mathcal{F}$ and would like to let $\mathcal{B}$ compute the value $\mathcal{F}(x)$ for an input x owned by $\mathcal{B}$. $\mathcal{A}$ learns nothing about x, and $\mathcal{B}$ learns only the computed result $\mathcal{F}(x)$ and does not gain any additional information about $\mathcal{F}$.

We use Oblivious Polynomial Evaluation [12, 4] to perform the set operations. $\mathcal{A}$ defines a polynomial W whose roots are her inputs $\mathcal{D}_{\mathrm{A}} = \{x_1, \ldots, x_n\}$:

$$W(y) = (y - x_1)(y - x_2)\ldots(y - x_n) = \sum_{u=0}^{n} \alpha_u y^u.$$

$\mathcal{A}$ encrypts the coefficients of this polynomial with the homomorphic public key cryptosystem and sends to $\mathcal{B}$ all the encrypted coefficients $\{E_{\mathrm{pk}}(\alpha_0), \ldots, E_{\mathrm{pk}}(\alpha_n)\}$, where $E_{\mathrm{pk}}(\alpha_i) = (g^{r_i}, h^{r_i} f^{\alpha_i})$. Because of the homomorphic properties, $\mathcal{B}$ can then evaluate the polynomial at the input β as:

$$E_{\mathrm{pk}}(W(\beta)) = (g^{\sum_{i=0}^{n} r_i\beta^i}, h^{\sum_{i=0}^{n} r_i\beta^i} f^{\sum_{i=0}^{n} \alpha_i\beta^i}),$$

and sends it to $\mathcal{A}$. When $\mathcal{A}$ receives $E_{\mathrm{pk}}(W(\beta))$, she decrypts it and obtains the information about $W(\beta)$.

2.3 Distributed Setting

The parties in our distributed model are a *Client* $\mathcal{C}$, a *Provider* $\mathcal{P}$, and w servers $S_1, S_2, \ldots, S_{\mathrm{w}}$. We assume that the provider holds a set of secrets $\mathcal{D}_{\mathrm{P}} = \{\mu_0, \mu_1, \ldots, \mu_{\mathrm{n\text{-}1}}\}$, which is distributed to w servers using (t, w)-Shamir's threshold secret sharing scheme. In addition, $\mathcal{P}$ sends $\lambda \overset{\mathrm{R}}{\leftarrow} \mathbb{Z}_{\mathrm{q}} - \{0\}$ to w servers. Suppose that there exits a public pseudo-random number generator G, whose outputs $G(\lambda)$ are uniformly distributed in $\mathbb{Z}_{\mathrm{q}} - \{0\}$. The notation $G(\lambda)$ means that the pseudo-random number generator G takes the random value λ.

$\mathcal{P}$ does not directly interact with $\mathcal{C}$ for the set operations, instead $\mathcal{C}$ contact at lest t servers to accomplish this task. Note, this distributed setting is not our contribution. Similar construction was proposed by Naor and Pinkas [13].

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

Initialization and Share Distribution Phase. First, $\mathcal{P}$ constructs a polynomial whose coefficients are his inputs (n secrets) as

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

Then $\mathcal{P}$ generates a random masking bivariate polynomial

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

such that $H(0,y)=0$ for any y. Using the polynomial H, $\mathcal{P}$ defines another bivariate polynomial $Q(x,y)=F(y)+H(x,y)$ satisfying $\forall y\ Q(0,y)=F(y)$. For $1\le \ell\le w$, $\mathcal{P}$ computes and sends $Q(\ell,y)$ to server S_ℓ as

$$Q(\ell,y)=F(y)+H(\ell,y)=\sum_{i=0}^{n-1}\mu_i y^i+\sum_{j=1}^{t-1}\sum_{i=0}^{n-1}a_{j,i}\ell^j y^i=\sum_{i=0}^{n-1}(\mu_i+b_{i,\ell})y^i.$$

Let $\mu_{i,\ell}=\mu_i+b_{i,\ell}$, the server S_ℓ receives a set of shares $\{\mu_{0,\ell},\ \dots,\ \mu_{n-1,\ell}\}$, which is the shared-coefficients [11] of the polynomial F.

Secret Reconstruction Phase. By Lagrange interpolation formula, we know that the coalition of t or more servers can reconstruct the original polynomial F. The t polynomials $Q(\ell_m,y)$ for $m\in[1,t]$ are:

$$\begin{aligned}Q(\ell_1,y)&=F(y)+\sum_{j=1}^{t-1}\sum_{i=0}^{n-1}a_{j,i}{\ell_1}^j y^i\\ &\ \ \vdots \qquad\qquad \vdots\\ Q(\ell_t,y)&=F(y)+\sum_{j=1}^{t-1}\sum_{i=0}^{n-1}a_{j,i}{\ell_t}^j y^i\end{aligned}$$

This can be written in matrix form as:

$$\begin{pmatrix}Q(\ell_1,y)\\Q(\ell_2,y)\\\vdots\\Q(\ell_t,y)\end{pmatrix}=\begin{pmatrix}1&\ell_1&{\ell_1}^2&\dots&{\ell_1}^{t-1}\\1&\ell_2&{\ell_2}^2&\dots&{\ell_2}^{t-1}\\\vdots&\vdots&\vdots&\ddots&\vdots\\1&\ell_t&{\ell_t}^2&\dots&{\ell_t}^{t-1}\end{pmatrix}\begin{pmatrix}F(y)\\\sum_{j=1}^{t-1}a_{j,0}\,y^i\\\vdots\\\sum_{j=1}^{t-1}a_{j,n-1}\,y^i\end{pmatrix}, \qquad (1)$$

where the coefficient matrix A is a Vandermonde matrix. As in the original (t,w)-Shamir scheme, the ℓ_m's are all distinct, so the matrix A is invertible [15] and consequently we obtain a unique solution in $\mathbb{Z}_q$. So if we multiply both sides of (1) by the inverse A^{-1}, we obtain

$$A^{-1}\begin{pmatrix}Q(\ell_1,y)\\Q(\ell_2,y)\\\vdots\\Q(\ell_t,y)\end{pmatrix}=\begin{pmatrix}F(y)\\\sum_{j=1}^{t-1}a_{j,0}\,y^i\\\vdots\\\sum_{j=1}^{t-1}a_{j,n-1}\,y^i\end{pmatrix}. \qquad (2)$$

Since we are interested in reconstructing F, we need only the first row of A^{-1}. Let $(v_{1,1},\ v_{1,2},\ \ldots,\ v_{1,t})$ be the first row of A^{-1}, Equation (2) can be simplified as:

$$\begin{pmatrix} v_{1,1} \; v_{1,2} \ldots v_{1,t} \end{pmatrix} \begin{pmatrix} Q(\ell_1, y) \\ \vdots \\ Q(\ell_t, y) \end{pmatrix} = F(y).$$

Therefore

$$\begin{pmatrix} v_{1,1} \; v_{1,2} \ldots v_{1,t} \end{pmatrix} \begin{pmatrix} \mu_{0,\ell_1} & \mu_{1,\ell_1} & \cdots & \mu_{n-1,\ell_1} \\ \vdots & \vdots & \ddots & \vdots \\ \mu_{0,\ell_t} & \mu_{1,\ell_t} & \cdots & \mu_{n-1,\ell_t} \end{pmatrix} = \begin{pmatrix} \mu_0 \; \mu_1 \ldots \mu_{n-1} \end{pmatrix}. \tag{3}$$

From (3) we conclude that the original set of secrets can be reconstructed as $\mu_i = \sum_{j=1}^{t} v_{1,j}\, \mu_{i,\ell_j}$ for $i \in [0, n-1]$.

Lemma 1. *Equation* (3) *for reconstructing the original secret is the same as the Lagrange interpolation formula and* $v_{1,j} = \prod_{\substack{1 \le k \le t \\ k \ne j}} \frac{\ell_k}{\ell_k - \ell_j}$ *for* $1 \le j \le t$.

Lemma 2. *In Lagrange's interpolation formula,*

$$\sum_{j=1}^{t} \left(\prod_{\substack{1 \le k \le t \\ k \ne j}} \frac{\ell_k}{\ell_k - \ell_j} \right) = 1.$$

We sketch the proofs of Lemmas 1 and 2 in Appendix A.

Note that we use the variant of ElGamal that is defined over $\mathbb{Z}_p$. We omit modulus p in the rest of the presentation, if no confusion will occur.

2.4 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 [6] for a more complete discussion.

In this model, there is no direct interaction between $\mathcal{C}$ and $\mathcal{P}$. $\mathcal{C}$ and w servers are assumed to act according to their prescribed actions in the protocol. The security definition is straightforward: (i) only $\mathcal{C}$ learns the result of the protocol, (ii) no server colludes with $\mathcal{C}$ to cheat.

Definition 1. (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 has no information about the provider's dataset and the set operation result.*

Following [12, 13] our model should meet the following requirements:

Correctness. A distributed protocol is correct if the client $\mathcal{C}$ obtains the result of evaluated shares of her input from t servers (and additional information from any single server in the second round if it is applicable), when each server with the shared-coefficients of the polynomial F and the client $\mathcal{C}$ with the input follow the steps of the protocol.

***Client*'s security.** Given that w servers get no output from the protocol, the definition of the client's privacy requires simply that any server cannot distinguish between cases in which the client has different inputs.

***Provider*'s security.** The definition ensures that the client does not get any extra information other than the output of the function. In addition, the provider's privacy should be *t-secure*.

3 Privacy-Preserving Distributed Set Intersection

Problem Definition: Let $\mathcal{P}$'s dataset be $\mathcal{D}_{\mathrm{P}} = \{\mu_0, \dots, \mu_{n-1}\}$. The construction and distribution of the initial shares and the random value λ are given in Section 2.3. Assume that $\mathcal{C}$ has a dataset $\mathcal{D}_{\mathrm{C}} = \{c_0, \dots, c_{m-1}\}$. We define the distributed set intersection problem as follows: $\mathcal{C}$ contacts the threshold of t or more servers to computes the intersection of $\mathcal{D}_{\mathrm{C}}$ and $\mathcal{D}_{\mathrm{P}}$ without gaining any other information about $\mathcal{D}_{\mathrm{P}}$. The contacted servers do not learn any information about $\mathcal{D}_{\mathrm{C}}$.

Corollary 1. *Let* $b_j = \prod_{\substack{1 \le k \le t \\ k \ne j}} \frac{\ell_k}{\ell_k - \ell_j}$ *in the interpolation formula for* $j \in [1, t]$. *Then* $\sum_{j=1}^{t} b_j (\mu_{i,\ell_j} - \mu_i) = 0$.

Proof. From Lemma 2, we know that $\sum_{j=1}^{t} b_j = 1$. We also know that $\sum_{j=1}^{t} \mu_{i,\ell_j} b_j = \mu_i$ by the interpolation formula. The conclusion is straightforward.

With the property in Corollary 1, we give a solution of distributed set intersection problem in Fig. 1.

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

Input: $\mathcal{C}$ has a set of data $\mathcal{D}_{\text{C}}$. $\mathcal{P}$ has another set of data $\mathcal{D}_{\text{P}}$, which is distributed to w servers. Each server S_ℓ knows the random value λ and the set of shares $\{\mu_{0,\ell}, \dots, \mu_{n-1,\ell}\}$ for $1 \leq \ell \leq w$.
Output: $\mathcal{C}$ knows $\mathcal{D}_{\text{C}} \cap \mathcal{D}_{\text{P}}$.

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

Fig. 1. Privacy-preserving distributed set intersection protocol

Efficiency. Regarding efficiency, the scheme requires only one round. $\mathcal{C}$ broadcasts a set of encrypted m secrets to t servers. Each contacted server S_{ℓ_j} responds with mn messages. So the communication complexity of this protocol is $O(tmn \times \log_2 p)$ bits.

For computation, $\mathcal{C}$ needs $m+2$ modular exponentiations and m multiplications for encrypting her dataset; tmn decryptions, tmn modular exponentiations and $mn(t-1)$ multiplications for the Lagrange interpolation. Each contacted server S_{ℓ_j} needs mn exponentiations and multiplications respectively for inputing its shares. So the computation complexity of this protocol is $O(tmn)$ multiplications considering that 1-exponentiation takes at most $\lfloor \log_2(p-1) \rfloor$ multiplications in Fast Exponentiation Algorithm.

Security. The above Distributed Set-Intersection protocol has following properties:

Theorem 1. *Assuming that the underlying homomorphic cryptosystem is semantically secure in the distributed set-intersection protocol, then any contacted server cannot distinguish any two inputs of $\mathcal{C}$.*

Proof. We need to show that for all $\upsilon \in [0, m-1]$, $E_{\text{pk}}(c_\upsilon)$'s look random if the DDH assumption holds. First, we choose $\upsilon' \in [0, m-1]$ where $\upsilon' \neq \upsilon$, and then $E_{\text{pk}}(c_\upsilon) = (g^{r_\upsilon}, h^{r_\upsilon} f^{c_\upsilon})$ and $E_{\text{pk}}(c_{\upsilon'}) = (g^{r_{\upsilon'}}, h^{r_{\upsilon'}} f^{c_{\upsilon'}})$. Since $r_\upsilon, r_{\upsilon'}$ are randomly picked from $\mathbb{Z}_{\text{q}}$, nobody can get the information about f^{c_υ} and $f^{c_{\upsilon'}}$without the private key. If the underlying homomorphic cryptosystem is semantically secure, we can conclude that $E_{\text{pk}}(c_\upsilon)$ and $E_{\text{pk}}(c_{\upsilon'})$ are indistinguishable.

Theorem 2. *Assuming that the discrete logarithm problem and the integer factorial problem are hard, $\mathcal{C}$ cannot compute any information other than $\mathcal{D}_C \cap \mathcal{D}_P$. In addition $\mathcal{P}$'s privacy is* t-secure.

Proof (Sketch). Assuming the discrete logarithm problem and the integer factorial problem are hard, $\mathcal{C}$ cannot infer anything from $f^{\lambda(\mu_{i,\ell_j}-c_\upsilon)}$ when it decrypts the ciphertext from each of t servers. By Corollary 1 if $f^{\lambda\sum_{j=1}^{t} b_j\,(\mu_{i,\ell_j}-c_\upsilon)} = 1$, then $c_\upsilon = \mu_i$, more precisely $c_\upsilon \in \mathcal{D}_\text{P}$; otherwise $f^{\lambda\sum_{j=1}^{t} b_j\,(\mu_{i,\ell_j}-c_\upsilon)}$ is a random integer. Therefore $\mathcal{C}$ cannot deduce anything from the output if $c_\upsilon \notin \mathcal{D}_\text{P}$. Furthermore, $\mathcal{C}$ cannot compute anything from less than t servers information. This is guaranteed by the polynomial secret sharing scheme we used.

4 Distributed Subset Relation

Problem Definition: Let $\mathcal{P}$'s dataset $\mathcal{D}_\text{P} = \{\mu_0, \dots, \mu_{n-1}\}$. The construction and distribution of the initial shares and the random value λ (with the public pseudo-random number generator G) are given in Section 2.3. Assume that $\mathcal{C}$ has another dataset $\mathcal{D}_\text{C} = \{c_0, \dots, c_{m-1}\}$. We define the distributed subset relation problem as follows: $\mathcal{C}$ learns if $\mathcal{D}_\text{C} \subseteq \mathcal{D}_\text{P}$ without gaining and revealing any other information.

To test subset relation, we apply the oblivious polynomial evaluation technique with an extra round of interaction between $\mathcal{C}$ and any of the w servers.

Correctness. In the protocol (Fig. 2), $\mathcal{C}$ first encrypts each element c_υ of her dataset by using her public key as $E_\text{pk}(f^{-c_\upsilon})$ for $\upsilon \in [0, m-1]$, and broadcasts all these encrypted elements to t servers. For every encrypted element $E_\text{pk}(f^{-c_\upsilon})$, each contacted server $S_{\ell_j}(1 \le j \le t)$ computes $E_\text{pk}(f^{\theta_\upsilon+(\mu_{i,\ell_j}-c_\upsilon)})$ for $0 \le i \le n-1$, and sends all the $E_\text{pk}(f^{\theta_\upsilon+(\mu_{i,\ell_j}-c_\upsilon)})$'s back to $\mathcal{C}$. $\mathcal{C}$ then decrypts those $E_\text{pk}(f^{\theta_\upsilon+(\mu_{i,\ell_j}-c_\upsilon)})$'s, and computes $d_{\upsilon,i} = f^{\theta_\upsilon\sum_{j=1}^{t} b_j+\sum_{j=1}^{t} b_j\,(\mu_{i,\ell_j}-c_\upsilon)}$ for each $i = 0, \dots, n-1$. Note that $\sum_{j=1}^{t} b_j = 1$ from Lemma 2, hence $d_{\upsilon,i} = f^{\theta_\upsilon+\sum_{j=1}^{t} b_j\,(\mu_{i,\ell_j}-c_\upsilon)}$. Then $\mathcal{C}$ defines a polynomial W by using all the $d_{\upsilon,i}$'s as its roots, and sends the coefficients of W to any of the w servers. Observe that if any $c_\upsilon = \mu_i$, then $d_{\upsilon,i} = f^{\theta_\upsilon+\sum_{j=1}^{t} b_j\,(\mu_{i,\ell_j}-c_\upsilon)} = f^{\theta_\upsilon}$. This means that $W(f^{\theta_\upsilon}) = 0$. By the homomorphic property, the only contacted server e in the second round is able to compute $E_\text{pk}(W(f^{\theta_\upsilon}))$. To save some computations, we only compute R_υ, the second part of $E_\text{pk}(W(f^{\theta_\upsilon}))$ for $\upsilon \in [0, m-1]$. Using all the R_υ's, server e then computes $\prod_{\upsilon=0}^{m-1} E_\text{pk}(W(f^{\theta_\upsilon}))$. When $\mathcal{C}$ receives and decrypts $\prod_{\upsilon=0}^{m-1} E_\text{pk}(W(f^{\theta_\upsilon}))$, it concludes $\mathcal{D}_\text{C} \subseteq \mathcal{D}_\text{P}$ if it is 1; otherwise $\mathcal{D}_\text{C}$ is not a subset of $\mathcal{D}_\text{P}$.

Efficiency. The protocol requires two rounds. First, $\mathcal{C}$ broadcasts a set of encrypted m secrets to t servers. Each contacted server S_{ℓ_j} responds with mn messages. In the second round, $\mathcal{C}$ sends $mn+1$ encrypted coefficients to one server and receives only one message. So the communication complexity of this protocol is still $O(tmn \times \log_2 p)$ bits.

Input: $\mathcal{C}$ has a set of data $\mathcal{D}_{\text{C}}$. $\mathcal{P}$ has another set of data $\mathcal{D}_{\text{P}}$, which is distributed to w servers. Each server S_ℓ knows the random value λ (with the public pseudo-random number generator G) and the set of shares $\{\mu_{0,\ell}, \dots, \mu_{n-1,\ell}\}$ for $1 \le \ell \le w$.
Output: $\mathcal{C}$ obtains 1 if $\mathcal{D}_{\text{C}} \subseteq \mathcal{D}_{\text{P}}$; otherwise a random integer.

1. Steps 1 - 3(b) are same as the distributed set-intersection protocol (Fig. 1), except one more step is added before step 2(a), and $\tau_{v,i,j}$ should be $(g^r, h^r f^{\theta_v + (\mu_{i,\ell_j} - c_v)})$ in step 2(a) accordingly.
2. (a$_0$) generates θ_v for $0 \le v \le m-1$ from $G(\lambda)$.
   (a) computes $\tau_{v,i,j} \leftarrow (g^r, h^r f^{\theta_v + (\mu_{i,\ell_j} - c_v)})$ for $v \in [0, m-1]$ and $i \in [0, n-1]$.
4. (a) $\mathcal{C}$ defines a polynomial W whose roots are $d_{v,i}$'s

$$\begin{aligned} W(y) &= (y - d_{0,0}) \dots (y - d_{0,n-1})(y - d_{1,0}) \dots (y - d_{m-1,n-1}) \\ &= \sum_{u=0}^{mn} a_u y^u. \end{aligned}$$

   (b) $\mathcal{C}$ randomly picks e from w and sends $\{\ E_{\text{pk}}(a_0), \dots, E_{\text{pk}}(a_{mn})\ \}$ to the server e, where $E_{\text{pk}}(a_u) = (g^{r'}, h^{r'} f^{a_u})$.
5. Server e
   (a) computes $R_v = h^{\sum_{j=0}^{mn} r' f^{j\theta_v}} f^{\sum_{j=0}^{mn} a_j f^{j\theta_v}}$ for $0 \le v \le m-1$.
   (b) computes $\hat{E} = \prod_{v=0}^{m-1} E_{\text{pk}}(W(f^{\theta_v})) = \left(g^{r' \sum_{v=0}^{m-1} \sum_{j=0}^{mn} f^{j\theta_v}}, \prod_{v=0}^{m-1} R_v \right)$.
   (c) sends $\hat{E}$ back to $\mathcal{C}$.
6. $\mathcal{C}$ decrypts $E_{\text{pk}}(\hat{E})$, and checks if it is 1.

Fig. 2. Privacy-preserving distributed subset relation protocol

For computation, $\mathcal{C}$ needs $m+2$ modular exponentiations and m multiplications for encrypting her own dataset; $tmn+1$ decryptions, tmn modular exponentiations and $(t-1)mn$ multiplications for the Lagrange interpolation; $2^{mn} - mn$ additions and $(mn-1)2^{mn} - mn + 1$ multiplications for polynomial expansion. Each contacted server S_{ℓ_j}, in the first round, needs mn exponentiations and multiplications respectively for inputing its shares. In the second round, the only contacted server e needs $m^2 n - m$ multiplications to precompute $f^{2\theta_v}, \dots, f^{mn\theta_v}$ for $1 \le v \le m$. Server e needs $m^2 n$ multiplications and exponentiations respectively for computing all R_v's. Denote $\hat{E}$ as $(\hat{L}, \hat{R})$, it needs $m(n+1) - 1$ additions and one exponentiation to compute $\hat{L}$ and m multiplications to get $\hat{R}$. So the computation complexity of this protocol is $O(2^{mn})$ additions and multiplications respectively using the Fast Exponentiation Algorithm.

Security. The security and its proof for this protocol is similar to that of the distributed set-intersection protocol.

Input: $\mathcal{C}$ has a set of data $\mathcal{D}_{\text{C}}$. $\mathcal{P}$ has another set of data $\mathcal{D}_{\text{P}}$, which is distributed to w servers. Each server S_ℓ knows the random value λ (with the public pseudo-random number generator G) and the set of shares $\{\mu_{0,\ell},\ \ldots,\ \mu_{n-1,\ell}\}$ for $1 \leq \ell \leq w$.
Output: $\mathcal{C}$ knows $|\mathcal{D}_{\text{C}} \cap \mathcal{D}_{\text{P}}|$.

1. Steps 1 - 3(b) are same as the distributed subset relation protocol (Fig. 2).
3. (c) $\mathcal{C}$ randomly picks e from w and sends $\{\ E_{\text{pk}}(d_{0,0}), \ldots, E_{\text{pk}}(d_{m-1,n-1})\ \}$ to the server e, where $E_{\text{pk}}(d_{\upsilon,i}) = (g^{r'}, h^{r'} d_{\upsilon,i})$.
4. Server e
   (a) computes
$$E_{\text{pk}}(d'_{\upsilon,i}) = \left(g^{\theta_\upsilon r'}, \left(h^{r'} d_{\upsilon,i} f^{-\theta_\upsilon}\right)^{\theta_\upsilon}\right)$$
$$= \left(g^{\theta_\upsilon r'}, h^{\theta_\upsilon r'} f^{\theta_\upsilon \sum\limits_{j=1}^{t} b_j\ (\mu_{i,\ell_j} - c_\upsilon)}\right)$$
   for $\upsilon \in [0, m-1]$ and $i \in [0, n-1]$.
   (b) shuffles all the $E_{\text{pk}}(d'_{\upsilon,i})$ and sends the shuffled $\{\widehat{d'_1}, \ldots, \widehat{d'_{mn}}\}$ back to $\mathcal{C}$.
5. (a) $\mathcal{C}$ decrypts $D_{\text{sk}}(\widehat{d'_\iota})$ and obtains 1 if one element of $\mathcal{D}_{\text{C}}$, which is carried in the computation of $\widehat{d'_\iota}$, is in $\mathcal{D}_{\text{P}}$; otherwise obtains a random integer.
   (b) $\mathcal{C}$ counts the number of ciphertexts received that decrypt to 1.

Fig. 3. Privacy-preserving cardinality of distributed set-intersection protocol

5 Variant: Cardinality of Distributed Set-Intersection

Problem Definition: Let $\mathcal{P}$'s dataset be $\mathcal{D}_{\text{P}} = \{\mu_0, \ldots, \mu_{n-1}\}$. The construction and distribution of the initial shares and the random value λ (with the public pseudo-random number generator G) are given in Section 2.3. Assume that $\mathcal{C}$ has another dataset $\mathcal{D}_{\text{C}} = \{c_0, \ldots, c_{m-1}\}$. We define the cardinality of distributed set-intersection problem as follows: $\mathcal{C}$ learns $|\mathcal{D}_{\text{C}} \cap \mathcal{D}_{\text{P}}|$ without gaining and revealing any other information.

By using oblivious polynomial evaluation, we need obtain the coefficients $a_0, \ldots, a_n$ of $W(y) = \sum_{i=0}^{n} a_i\, y^i$ from the factorization $\prod_{i=0}^{n-1}(y - c_i)$. To avoid this cost, we simply use the technique of random shuffling to permute the ordered encrypted results in the second round by the only contacted server.

The protocol, given in Fig. 3, proceeds the same as the distributed subset relation protocol until Step 3(b). $\mathcal{C}$ then encrypts each of the $d_{\upsilon,i}$'s and sends all the $E_{\text{pk}}(d_{\upsilon,i})$'s to any of the w servers. The only contacted server multiplies $f^{-\theta_\upsilon}$ to the second part of $E_{\text{pk}}(d_{\upsilon,i})$, then raises the power of θ_υ to both components of $E_{\text{pk}}(d_{\upsilon,i})$. When this process concludes, the only contacted server randomly permutes all the ciphertexts and returns them to $\mathcal{C}$. $\mathcal{C}$ counts the number of ciphertexts received that decrypt to 1. The security proof for this protocol trivially follows that of the distributed subset relation protocol.

The efficiency is the same as the distributed subset relation protocol for the first round. In the second round of this protocol, $\mathcal{C}$ sends mn messages to the server e and the server e sends mn messages back. For the computation, $\mathcal{C}$ performs mn encryptions and decryptions respectively in the second round. Accordingly, the server e performs mn multiplications and mn modular exponentiations.

6 Conclusion and Future Work

In this paper, we have proposed a protocol for the privacy-preserving distributed set intersection problem by combining Shamir's secret sharing scheme and homomorphic encryption scheme. Moreover, we have shown that with a second round of interaction and a random shuffling the cardinality of distributed set-intersection could be computed efficiently. By using oblivious polynomial evaluation techniques, we have also constructed an two-round privacy-preserving distributed subset relation protocol.

The further research will be to provide a solution of above distributed set operations against active adversary.

References

[1] 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.

[2] 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.

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

[4] 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.

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

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.
[13] 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.
[14] 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.
[15] D. R. Stinson. An explication of secret sharing scheme. *Designs, Codes and Cryptography*, 2:357–390, 1992.
[16] 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.
[17] 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 and 2

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

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

Since the points are pairwise distinct, A is invertible. Let $A^{-1} = (v_{i,j})_{1 \le i,j \le t}$. By taking first row of A^{-1} and first column of A, we obtain $\sum_{j=1}^{t} v_{1,j} = 1$.

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

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

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

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

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

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

Let $A \times D = \mathcal{W} = (w_{\varsigma,\eta})_{\substack{1 \le \varsigma \le t \\ 1 \le \eta \le t}}$. To fix $\varsigma, \eta \in [1,t]$ the coefficient $w_{\varsigma,\eta}$ is obtained by using the ς^{th} row of A 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 $A^{-1} = D$.

Since the sum of the first row of A^{-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 $\forall j \in [1,t],\ a_{j,1} = P_j(0) = \prod_{\substack{1\le k\le t \\ k\neq j}} \frac{x_{i_k}}{x_{i_k} - x_{i_j}}$. Combining the previous two findings, we can conclude that:

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