Constant-Round Privacy Preserving Multiset Union

Jeongdae Hong[1], Jung Woo Kim[1], Jihye Kim[2], Kunsoo Park[1], and Jung Hee Cheon[2]

[1] School of Computer Science and Engineering, Seoul National University
{jdhong,jkim,kpark}@theory.snu.ac.kr
[2] ISaC and Department of Mathematical Sciences, Seoul National University
{jihyek,jhcheon}@snu.ac.kr

Abstract. Privacy preserving multiset union (PPMU) protocol allows a set of parties, each with a multiset, to collaboratively compute a multiset union *secretly*, meaning that any information other than union is not revealed. We propose efficient PPMU protocols, using multiplicative homomorphic cryptosystem. The novelty of our protocol is to directly encrypt a polynomial by representing it by an element of an extension field. The resulting protocols consist of constant rounds and improve communication cost. We also prove the security of our protocol against malicious adversaries, in the random oracle model.

Key words: Privacy Preserving Multiset Union, ElGamal on Polynomials, Homomorphic Encryption

1 Introduction

Privacy Preserving Multiset Union (PPMU) is a set operation that a set of parties, each with a multiset, to collaboratively compute a multiset union but no party learns more information about other parties' private inputs than what can be deduced from the result of union.

PPMU is useful in various applications such as data collection for statistics and majority voting, where each element is related with privacy or interests of the data owner. For example, a research group wants to collect information about patients with sparse disease, while hospitals or patients want to protect their privacy. In the Rawlings Gold Glove Award, managers and coaches select the player judged to have the most superior individual fielding performance at each position by voting. A company wants to collect clients' claims to provide better services for its clients, who are reluctant to directly publish their claims for their privacy.

In 1982, Yao proposed a method for a millionaire's problem [23], which is the first privacy preserving multiparty computation and can be used for other problems like secret voting, private querying of database, oblivious negotiation, playing mental poker, etc. Various solutions for millionaire's or private equality test problem have been proposed [1, 18, 14]. In addition, other types of multiparty computations have followed such as set intersection [6, 16, 5, 10, 20], set element reduction [12], and set union [12, 11, 2, 7]. However, PPMU itself has been paid comparatively less attention though electronic voting and shuffling algorithms solve the similar problem.

PPMU is firstly addressed by Kissner and Song (KS) in [12]. In fact, KS proposed a more general version of PPMU, i.e., threshold set union protocol which computes the elements which appear at least a threshold number of times in the parties private inputs.

By setting the threshold to be one in KS threshold set union, we can obtain a PPMU protocol based on a *shuffling* protocol on their private inputs. For shuffling, KS consider either a mix-net [8, 15] or their shuffling protocol. If a mix-net is used, all parties should perform shuffling in turn. Thus, the number of rounds increases linearly in the number of parties. KS also proposed their own shuffling protocol, which is more efficient than using a mix-net. Still, since the shuffling protocol by KS also relays an intermediate shuffling result to the other party in turn, their solution does not improve the round complexity itself. The shuffling protocol by KS relies on an additively homomorphic cryptosystem over an *unknown* order group with an efficient factoring algorithm. However, such a group has not been known yet. Even if such a cryptosystem exists, their protocols have linear round complexity. It is an interesting problem whether we can construct a constant-round PPMU protocol, for example, by allowing each party to parallel their protocol execution.

Assuming a broadcast medium, we propose constant-round privacy-preserving multiset union (PPMU) protocols based on multiplicatively homomorphic cryptosystem. Given a polynomial with roots of inputs, the novelty of our protocol is to directly encrypt the polynomial by embedding it into the base field element, instead of encrypting each coefficient of the polynomial. Given encryptions on multiplicatively homomorphic cryptosystem, each party can multiply all encryptions in *parallel*, resulting into an encryption of the set union. In other words, it produces a constant-round protocol. As a final step, we need to factorize the polynomial associated to the set union. It is supported by ElGamal cryptosystem on an extension field which defines a message over a field with *known* order.

Another interesting aspect is that our broadcast-based protocol can be adapted into a unicast-based protocol, *without* increasing its message size. The unicast-based protocol relays its message, accumulating party's input elements sequentially. If the encryption of a polynomial is represented as the encryption of related coefficients then it seems to be avoidable that each relay expands the size of the encrypted message as in the KS protocol. Since our protocol represents the encryption of a polynomial as an element of the finite field, accumulation does not incur any additional message overhead. In fact, whether to use the broadcast medium or the unicast medium, the size of the encrypted message is the same in our scheme. Thus, our scheme can be efficiently implemented in both broadcast and unicast environments.

Contributions of our work are summarized as follows:

1. We propose constant-round privacy-preserving multiset union (PPMU) protocols. To the best of our knowledge, our PPMU protocol is the *first* result with a constant-round complexity. (Previous works [12, 7] have linear round complexity.)
2. We improve the communication size. The total number of bits transmitted in our protocol is $n^2 k \log q$, while that in KS's protocol is $n^2 k \log N$, where n is the number of parties, k the number of elements and q the size of the element domain. Therefore, our protocol has the $\frac{\log q}{\log N}$ times smaller communication size than KS's. (For the details, refer to the section 5.)
3. To obtain computational efficiency gains, we revise ElGamal encryption so that it has short-size exponents. The revised ElGamal is semantically secure in the short interval decisional Diffie-Hellman assumption of which security is proved in the generic model.
4. Finally, we prove security of proposed protocols, in presence of malicious adversaries in random oracle model. In particular, we efficiently add commitment and standard zero-knowledge proofs to construct a protocol secure against malicious adversaries, without increasing asymptotic complexities in comparison to the protocol secure against honest-but-curious adversaries.

Organization. The rest of the paper is organized as follows. Section 2 gives some preliminary information. We propose ElGamal on polynomials in Section 3. We present PPMU protocols using ElGamal on polynomials in Section 4 and compare the performance of our PPMU protocol with that of the KS scheme in Section 5. In Section 6, we conclude the paper.

2 Preliminaries

2.1 Adversary Models

We consider two standard adversary models: honest-but-curious adversaries and malicious adversaries. The security definition is limited to the case where at least one of the parties is honest. We describe informal definitions of these models; formal definitions of these models can be found in [9].

Honest-But-Curious Adversary. In this model, all parties act according to their prescribed actions in the protocol. Security in this model is as follows: no party or coalition of cheating parties who share their private information gains information about players' private input sets other than what can be deduced from the result of the protocol. This is formalized by considering an ideal implementation where a trusted third party (TTP) receives the inputs of the parties and outputs the result of the defined function. We require that in the real implementation of the protocol (that is one without a TTP) each party does not learn any information other than that in the ideal implementation.

Malicious Adversary. In this model, an adversary may behave arbitrarily. The standard security definition in this model captures correctness and privacy issues of the protocol. The security of a protocol is analyzed by comparing what an adversary can do in a real protocol execution to what it can do in an ideal scenario (that is secure by definition). This is formalized by considering an ideal computation involving a trusted third party to whom the parties send their inputs. The TTP computes the functionality on the inputs and returns to each party its respective output. Informally, a protocol is secure if no adversary interacting in the real protocol (where no TTP exists) can do more harm in a real execution than in an execution that takes place in the ideal world. In other words, for any adversary carrying out a successful attack on a real protocol, there exists a simulator that successfully carries out the same attack in the ideal world.

2.2 Techniques and Tools

Polynomial Representation of Sets. We represent a set of elements by a polynomial. For example, given a set of k elements $S = \{m_1, m_2, \cdots, m_k\}$ where each m_i is an element of $\mathbb{F}_q$, we construct its polynomial representation as

$$M(t) = \prod_{i=1}^{k}(t - m_i)$$

Multiset Union using Polynomial Representations. We follow Kissner and Song's representation [12] to compute multiset union. Let p_1 and p_2 be polynomial representations of multisets S and T, respectively. They defined the union $S \cup T$ as a multiset, where each element m_i that appears $n_S \geq 0$ times in S and $n_T \geq 0$ times in T appears in the resulting multiset $n_S + n_T$ times. Then, one can compute the polynomial representation of $S \cup T$ as

$$p_1 * p_2.$$

Note that $p_1 * p_2$ is a polynomial representation of $S \cup T$ because all elements that appear in either set S or T are preserved and duplicate elements from each multiset are preserved. In addition, given the polynomial representation of $S \cup T$, one cannot learn any information about S and T other than what can be deduced from $S \cup T$.

Multiplicative Homomorphic Cryptosystem. In this paper we utilize a semantically secure, multiplicative homomorphic public-key cryptosystem. Namely, given the encryption of m_1 and m_2, $E_{pk}(m_1)$ and $E_{pk}(m_2)$, one can efficiently compute the encryption of $m_1 \cdot m_2$ denoted by $E_{pk}(m_1 \cdot m_2)$. We also require that the homomorphic public-key cryptosystem support secure (n, n)-threshold decryption, i.e., the corresponding private key is shared by a group of n players, and decryption must be performed by *all* players acting together.

In the protocols for the malicious case, we require the decryption protocol be secure against malicious players (typically, this is done by requiring each party to prove by zero-knowledge that he has followed the threshold decryption protocol correctly) and efficient construction of zero-knowledge proofs of plaintext knowledge.

Note that the semantically secure version of ElGamal cryptosystem satisfies each of our requirements: it is multiplicative homomorphic, supports threshold decryption (secure in the malicious case) [4], and allows the zero knowledge proof of plaintext knowledge. (Refer to the next section for the details.)

3 ElGamal on Polynomials

3.1 Overview

We revisit the ElGamal cryptosystem and modify it in the following three steps:

- First, we modify the ElGamal cryptosystem so that it can directly encrypt a polynomial element, particularly in $(\mathbb{F}_q[t]/f(t))^*$ for some irreducible polynomial $f(t)$. Note that the general approach to encrypt a polynomial is to encrypt its coefficients (rather than encrypt the polynomial itself). This new encryption method supports efficient computation for multiset union encryption through its multiplicative homomorphic property.
- Second, we convert the basic ElGamal scheme to be semantically secure under the Decisional Diffie-Hellman (DDH) assumption. We limit all group operations to a subgroup where the DDH assumption holds. The semantic security of the ElGamal system is clear if the underlying cryptographic assumption, i.e., DDH assumption, holds.
- Third, we modify the semantically secure ElGamal scheme such that the exponent size is limited to a short interval, for example, of 160 bits. This modification is made for performance improvement. For security of our modified ElGamal cryptosystem, we propose a new variant of the DDH assumption called the Short Interval DDH (SI-DDH) assumption and prove the security of SI-DDH in the generic model.

In the following sections, we describe the cryptographic assumptions and our schemes.

3.2 Cryptographic Setting

Let $\mathbb{G}$ be a cyclic group of prime order q and g be its generator.

Decisional Diffie-Hellman (DDH) Problem. The DDH problem in $\mathbb{G}$ is as follows: Given $g, g^x, g^y, g^z \in \mathbb{G}$, where x, y, z are randomly chosen from $\mathbb{Z}_q$, determine whether $g^z = g^{xy}$ or not.

DDH Assumption. The DDH problem is (ϵ, t)-hard in $\mathbb{G}$ if for every algorithm A running in time t we have:

$$\begin{aligned} | \Pr[x, y \leftarrow \mathbb{Z}_q : A(g, g^x, g^y, g^{xy}) = 1] - \\ \Pr[x, y, z \leftarrow \mathbb{Z}_q : A(g, g^x, g^y, g^z) = 1] \ | \ \leq \ \epsilon \end{aligned}$$

Informally, we say that DDH holds in $\mathbb{G}$ if ϵ is negligible for all efficient algorithms A.

We introduce a new variant of DDH. For clarity of our description, we describe the corresponding discrete log and computational Diffie-Hellman problems as well.

Denote by $\mathbb{T}$ the subset of $\mathbb{G}$ generated by g^x with $x \in [1, 2^\ell]$ where ℓ is defined by the security parameter.

Short Interval Discrete Log (SI-DL) Problem. The SI-DL problem in $\mathbb{T}$ is as follows: Given $g, g^x \in \mathbb{T}$, where x is randomly chosen from $[1, 2^\ell]$, compute x.

Short Interval Computational Diffie-Hellman (SI-CDH) Problem. The SI-CDH problem in $\mathbb{T}$ is as follows: Given $g, g^x, g^y \in \mathbb{T}$, where x, y are randomly chosen from $[1, 2^\ell]$, compute g^{xy}.

Short Interval Decisional Diffie-Hellman (SI-DDH) Problem. The SI-DDH problem in $\mathbb{T}$ is as follows: Given $g, g^x, g^y, g^z \in \mathbb{T}$, where x, y, z are randomly chosen from $[1, 2^\ell]$, determine whether $g^z = g^{xy}$ or not.

SI-DDH Assumption. The SI-DDH problem is (ϵ, t)-hard in $\mathbb{T} \subset \mathbb{G}$ if for every algorithm A running in time t we have:

$$\begin{aligned} &|\Pr[x, y \leftarrow [1, 2^\ell] : A(g, g^x, g^y, g^{xy}) = 1] - \\ &\Pr[x, y \leftarrow [1, 2^\ell], z \leftarrow [1, 2^{2\ell}] : A(g, g^x, g^y, g^z) = 1]| \leq \epsilon \end{aligned}$$

Informally, we say that SI-DDH holds in $\mathbb{T}$ if ϵ is negligible for all efficient A.

Hardness of SI-DDH. The best known attack for SI-DDH is Pollard's lambda algorithm [19]. Given g, g^x in a cyclic group $\mathbb{G}$ of order q, the Pollard's lambda algorithm allows us to search for x in some subset $\{a, \cdots, b\}$ of $\mathbb{Z}_q$. Note that we may search the entire range of possible logarithms by setting $a = 0$ and $b = q - 1$, although in this case Pollard's rho algorithm is more efficient. To disprove SI-DDH we can use a *generic* DDH algorithm. The best known generic DDH algorithm is a generic discrete log algorithm [3, 21]. When we apply this algorithm to a group of prime order q, the algorithm runs in time $\Omega(\sqrt{q})$. When we apply the generic discrete log algorithm to a group subset $\mathbb{T}$ of size 2^ℓ, the algorithm performs $\Omega(\sqrt{2^\ell})$ group operations. The proof of the following theorem is slightly modified from the one in [3].

Theorem 1. *$\Omega(\sqrt{2^\ell})$ is a lower bound on the complexity of any generic algorithm for the discrete logarithm problem in a subset $\mathbb{T}$ of size 2^ℓ.*

Proof. An *encoding* of $\mathbb{Z}_q$ on $\mathbb{X}$ is an injective mapping σ from $\mathbb{Z}_q$ into $\mathbb{X}$, where $\mathbb{X}$ is a set of bit strings of cardinality at least q. A generic Las Vegas algorithm, say *GenLog*, for $(\mathbb{Z}_q, +)$ on $\mathbb{X}$ is a probabilistic algorithm whose input consists of $\sigma_1 = \sigma(1)$ for a generator and $\sigma_2 = \sigma(x)$ for a value x, where $x \in [1, 2^\ell]$. *GenLog* will be successful if and only if it outputs the value x. *GenLog* will use the oracle to generate a sequence of w which is the encoding of linear combinations of 1 and x. Note that the only difference from [3] is that x is an integer in $[1, 2^\ell]$, not in $[0, q - 1]$. We follow the definition of $GOOD(C)$ in [3], and the probability that $x \in GOOD(C)$ is at most $\binom{w}{2}/2^\ell$. If $x \notin GOOD(C)$, then the best strategy for *GenLog* is to guess the value of x by choosing a random value in $[1, 2^\ell] \setminus GOOD(C)$. Let $g = |GOOD(C)|$. Then, we can compute a bound on the success probability of the algorithm. Suppose we define A to be the event $x \in GOOD(C)$ and we let B denote the event "the algorithm returns the correct value of x". Then we have

$$\begin{aligned}\Pr[B] &= \Pr[B|A] \times \Pr[A] + \Pr[B|\overline{A}] \times \Pr[\overline{A}] \\ &= 1 \times \frac{g}{2^\ell} + \frac{1}{2^\ell - g} \times \frac{2^\ell - g}{2^\ell} \\ &\le \frac{\binom{w}{2} + 1}{2^\ell}.\end{aligned}$$

If the algorithm always gives the correct answer, then $\Pr[B] = 1$. In this case, it is easy to see that w is $\Omega(\sqrt{2^\ell})$. □

3.3 Basic ElGamal on Polynomials

We employ ElGamal cryptosystem whose elements are in $(\mathbb{F}_q[t]/f(t))^*$, where $f(t)$ is an irreducible polynomial of degree d. We call this cryptosystem *ElGamal on Polynomials*. Thus we consider $M(t) = \prod_{i=1}^{k}(t - m_i)$, where $m_i \in \mathbb{F}_q$ and $k < d$, as an element of multiplicative cyclic group $(\mathbb{F}_q[t]/f(t))^*$ of order $q^d - 1$, and we can encrypt $M(t)$ as in the original ElGamal encryption.

Let $g(t)$ be a generator of the group $(\mathbb{F}_q[t]/f(t))^*$, and $y(t) = g(t)^x \bmod f(t)$, where x is a secret key randomly selected from $[0, q^d - 2]$; the polynomials $f(t)$, $g(t)$, and $y(t)$ are public keys. Then we can encrypt $M(t)$ with a random $r \in_R [0, q^d - 2]$ as

$$C(t) = (u(t), v(t)) = (g(t)^r, y(t)^r \cdot M(t)) \bmod f(t).$$

Note that, in our scheme, we can encrypt input set $\{m_1, \cdots, m_k\}$ at a time, since the set is represented as a polynomial, i.e., an element in $(\mathbb{F}_q[t]/f(t))^*$ which is the unit of encryption. Decryption can also be done as follows:

$$\frac{v(t)}{u(t)^x} \bmod f(t) = \prod_{i=1}^{k}(t - m_i).$$

The correctness of the scheme is obvious because we restrict the number of elements in S to be less than d which is the degree of $f(t)$. However, ElGamal on polynomials is not semantically secure since the DDH assumption does not hold in the group $\mathbb{F}_{q^d}^*$. Using the Euler's criterion[3], it is easy to test elements of $\mathbb{F}_{q^d}^*$ whether they are quadratic residues or not. Thus, it can be detected whether $M(t)$ is a quadratic residue or not by checking the quadratic residuosity of $v(t)$ and $y(t)^r$.

Adding Semantic Security. For semantic security we limit all group operations to a subgroup, i.e., quadratic residues, where the DDH assumption holds. Let $q^d - 1 = ap$ for some prime p and an even number a. To make ElGamal on polynomials semantically secure, we consider a subgroup of $(\mathbb{F}_q[t]/f(t))^*$ which satisfies DDH assumption:

$$\mathbb{G}_p = \{x(t)^{(q^d-1)/p} : x(t) \in (\mathbb{F}_q[t]/f(t))^*\} = \{x(t)^a : x(t) \in (\mathbb{F}_q[t]/f(t))^*\}$$

$$\mathbb{G}_p = \{(g(t)^0)^a, (g(t)^1)^a, \cdots, (g(t)^{p-1})^a\}$$

[3] For $\forall a(x) \in \mathbb{F}_{q^d}^*, a(x)$ is a quadratic residue $\bmod f(x)$ if and only if $a(x)^{(q^d-1)/2} \equiv 1 \bmod f(x)$

Since $(g(t)^p)^a = g(t)^{q^d-1} = 1 = g(t)^0$, this group is closed and it has an order of p. The generator of this group is $g(t)' = g(t)^{\frac{q^d-1}{p}}$, where $g(t)$ is the generator of $(\mathbb{F}_q[t]/f(t))^*$.

ElGamal on polynomials in $\mathbb{G}_p$ is defined by rearranging $u(t)$ and $y(t)$ as follows: $u(t) = g(t)'^r$ and $y(t) = g(t)'^x \bmod f(t)$, that is, $u(t)$ and $y(t)$ are generated from the generator of $\mathbb{G}_p$. Finally, $M(t)$ needs to be in $\mathbb{G}_p$. We convert $M(t) = \prod_{i=1}^{k}(t - m_i)$ into an element $M'(t)$ of $\mathbb{G}_p$ by combining proper random number R_i such that $M'(t) = \prod_{i=1}^{k}(t - m_i')$ where $m_i' = m_i||R_i$.

Then we can encrypt $M'(t)$ with random $r \in_R [0, p-1]$

$$C(t) = (u(t), v(t)) = (g(t)'^r, y(t)^r \cdot M'(t)) \bmod f(t)$$

where $y(t) = g(t)'^x$. Decryption can also be done as follows:

$$\frac{v(t)}{u(t)^x} \bmod f(t) = \prod_{i=1}^{k}(t - m_i').$$

we obtain the multiset $\{m_1, \cdots, m_k\}$ by factoring $f(t)$ and removing R_i's.

Remark. The step in encryption converting the polynomial into an element of the group $\mathbb{G}_p$ is expected to finish quickly. Since we select a subgroup $\mathbb{G}_p$ of size p and $q^d - 1 = ap$, the group $(\mathbb{F}_q[t]/f(t))^*$ is divided into a subgroups. Therefore, the step is expected to finish in a repetitions. Also, there exists a sufficiently small even number a for almost all q; and we confirmed it for a prime $3 \le q < 100$ through experiments. For example,

$3^{1091}-1 = 2 \cdot$ 17308478920290520561214081551955057184506477986582966242130879304040855943078281750655654
65877316796954389481170318472442241905892654388323782720732502237578480086224100629849484070049079760101
44446727825839493126338747635054243392729455736743049576570977803426383945195972613488488758512339944888
43015578069982923778120456915440994423370412559032927349367947178856907461526675259289295907986943204438
70675144426955467776443389120197038417789600161987266227110204849862069798191865497069657127372102640237
5168960030732173.

Knowledge of Plaintext. Note that given ElGamal encryption $C(t) = (u(t), v(t)) = (g(t)'^r, y(t)^r \cdot M'(t))$ with public key $y(t)$ the knowledge of $M_i'(t)$ corresponds to the knowledge of r: the knowledge of r extracts $M'(t) = v(t)/y(t)^r$.

Definition 1 (IND-CPA). *An encryption scheme is semantically-secure in terms of indistinguishability if no adversary, given an encryption of a message randomly chosen from a two-element message space determined by the adversary, can identify the message choice with probability significantly better than that of random guessing (1/2).*

Theorem 2. *The ElGamal scheme above is semantically secure against chosen plaintext attack (IND-CPA) under the DDH assumption.*

Proof. Assuming the DDH problem is (ϵ, t)-hard in $\mathbb{G}_p$, for any algorithm A' running in time t with $x, y, z \leftarrow \mathbb{Z}_q$ we have

$$|\Pr[A'(g, g^x, g^y, g^{xy} \cdot\ m_1) = 1] - \Pr[A'(g, g^x, g^y, g^z \cdot\ m_1) = 1]| \le\ \epsilon \tag{3.1}$$

by a reduction from the DDH problem. Note that we obtain the problem 3.1 by reducing DDH problem. Similarly again, assuming the DDH problem is (ϵ, t)-hard in $\mathbb{G}_p$, for any algorithm A' running in time t with $x, y, z \leftarrow \mathbb{Z}_q$ we have

$$|\Pr[A'(g, g^x, g^y, g^{xy} \cdot\ m_2) = 1] - \Pr[A'(g, g^x, g^y, g^z \cdot\ m_2) = 1]| \le\ \epsilon. \tag{3.2}$$

Since multiplying a random element by a fixed element yields a random element, for $x, y, z \leftarrow \mathbb{Z}_q$ we have

$$\Pr[A'(g, g^x, g^y, g^z \cdot\ m_1) = 1] = \Pr[A'(g, g^x, g^y, g^z \cdot\ m_2) = 1]. \tag{3.3}$$

¿From equations 3.1, 3.2, and 3.3, for $x, y \leftarrow \mathbb{Z}_q$ we have

$$|\Pr[A'(g, g^x, g^y, g^{xy} \cdot\ m_1) = 1] - \Pr[A'(g, g^x, g^y, g^{xy} \cdot\ m_2) = 1]| \leq\ 2\epsilon.$$

Thus, the ElGamal scheme in the group $\mathbb{G}_p$ is indistinguishable under chosen plaintext attack (IND-CPA). □

3.4 ElGamal on Polynomials with short exponents

When the ElGamal encryption is implemented in the group $\mathbb{G}_p$, one normally uses a $\log p$-bit exponent. Depending on applications, however, the time for modular exponentiation by a $\log p$-bit exponent can be burdensome. Since the discrete logarithm problem in $\mathbb{G}_p$ is infeasible by the Index Calculus Algorithm, one may consider the use of a subgroup of order p much smaller than $q^d - 1$ to speed up the encryption. For example, one may select the order p of only 160 bits long, while $q^d - 1$ is 1024 bits long. In this case, the ElGamal encryption is secure. However, since the size of subgroup $\mathbb{G}_p$ is extremely small compared to that of group $(\mathbb{F}_q[t]/f(t))^*$, the trial to convert a polynomial in $(\mathbb{F}_q[t]/f(t))^*$ to an element of group $\mathbb{G}_p$ by adding randomness will succeed with a very small probability ($\approx 1/2^{864}$).

Therefore, we take a different approach and propose a semantically secure ElGamal on polynomials with a short exponent in the subgroup $\mathbb{G}_p$ of order p, where $p \approx q^d - 1$ and the exponent size is 160 bits (assuming the 80-bit security level). This scheme is semantically secure under the SI-DDH assumption. Its full description is in Figure 1. Note that the exponents, x and r, are selected from a 160-bit short interval: other elements (i.e., $g(t), M'(t)$) are selected in $\mathbb{G}_p$.

Theorem 3. *The ElGamal scheme in Figure 1 is semantically secure against chosen plaintext attack (IND-CPA) under the SI-DDH assumption.*

Proof. The proof is the similar to that of Theorem 2. The difference is that the SI-DDH assumption is used and the domains for x, y, z are accordingly adjusted as: $x, y \in_R [1, 2^\ell]$ and $z \in_R [1, 2^{2\ell}]$.

4 Privacy Preserving Multiset Union (PPMU) Protocol Using ElGamal on Polynomials

First, we propose a protocol for the honest-but-curious model. Then, we consider a possible attack and its counter-measure, and extend our honest-but-curious protocol to the malicious model.

4.1 Protocol for the Honest-But-Curious (HBC) Model

Let S_i be the input set of party P_i $(1 \leq i \leq n)$ and $(S_i)_j$ be the j-th element of set S_i. Assume that $|S_i|$ for all i is k and $d > nk$. Our protocol allows all the parties to learn the union $S_1 \cup S_2 \cup \cdots \cup S_n$ with multiplicity, but the parties do not know which set an element comes from. The secret key x corresponding to the public key is shared to all the parties.

Let ℓ be defined by a security parameter and k be the number of elements. Let S be a multiset $\{m_1, \cdots, m_k\}$ where $m_i \in \mathbb{Z}_q$ and $k < d$.

KeyGeneration (ℓ, d)

1. Select q and d such that $k < d$ and $p|(q^d - 1)$, where q and p are primes and $q^d - 1 = ap$.
2. Select an irreducible polynomial $f(t) \in \mathbb{F}_q[t]$ with degree d and a generator $g(t)$ from $\mathbb{G}_p$.
3. Randomly choose a secret key $x \in_R [0, 2^\ell]$ and compute $y(t) = g(t)^x \bmod f(t)$.

Then $\mathcal{K} = \{(f(t), g(t), x, y(t)) : y(t) \equiv g(t)^x \bmod f(t)\}$. The polynomial $f(t)$, $g(t)$, and $y(t)$ are public keys (pk), and an integer x is a secret key.

Encryption (S, pk)

Randomly select R_i until $M'(t) = \prod_{i=1}^{k}(t - m'_i)$ is in $\mathbb{G}_p$, where $m'_i \leftarrow m_i||R_i$.

$$\mathcal{E}_{pk}(M'(t)) = (u(t), v(t)) = (g(t)^r, y(t)^r \cdot M'(t)) \bmod f(t)$$

where $r \in_R [0, 2^\ell]$.

Decryption $(u(t), v(t), x)$

$$v(t)/u(t)^x \bmod f(t) = \prod_{i=1}^{k}(t - m'_i)$$

Obtain the multiset $\{m_1, \cdots, m_k\}$ by factoring $f(t)$ and removing R_i's.

Fig. 1. Semantically secure ElGamal under the SI-DDH assumption

Define a key set

$$\mathcal{K} = \{(f(t), g(t), x, y(t)) : y(t) \equiv g(t)^x \bmod f(t)\} \tag{4.1}$$

where $g(t), y(t)$ are public keys and x is a private key.

In encryption phase, each party P_i calculates a polynomial $M'_i(t) = (t-(S_i)'_1) \cdots (t-(S_i)'_k)$ where $(S_i)'_j \leftarrow (S_i)_j||R_{ij}$ for random R_{ij} until $M'_i(t)$ belongs to $\mathbb{G}_p$. Each party encrypts it with a random $r_i \in_R [0, 2^\ell]$ as

$$C_i(t) = (g(t)^{r_i},\ y(t)^{r_i} \cdot M'_i(t)) \bmod f(t) \tag{4.2}$$

and broadcasts it to all other parties. Then, every party simultaneously can compute the encryption of multiplication of polynomials using the multiplicative homomorphic property of ElGamal cryptosystem.

$$C(t) = (g(t)^{r_1+\cdots+r_n},\ y(t)^{r_1+\cdots+r_n} \cdot \prod_{i=1}^{n} M'_i(t)) \tag{4.3}$$

Note that total degree nk of product of polynomials should be less than the degree d of polynomial $f(t)$.

In decryption phase, for a given ciphertext $C(t)$ all the parties perform a group decryption (i.e., (n, n)-threshold decryption) [4] to obtain product of polynomials. Then, all the parties learn all of the elements in the polynomial by using polynomial factoring. Using a square-free decomposition and the Cantor-Zassenhaus algorithm [22], it is possible to factor polynomials over finite fields. Also, it runs in expected polynomial time.

Note that this scheme can be easily combined with a threshold ElGamal cryptosystem to provide a PPMU scheme without TTP [17].

Security Analysis The only information that an honest-but-curious adversary can obtain is the encryptions of the sets of other parties or PPMU. Therefore, the adversary cannot obtain any element of other parties since the ElGamal on polynomials is semantically secure.

4.2 Protocol for the Malicious (MAL) Model

Protocol Malicious Protocol for PPMU

Input: There are $n > 2$ parties P_i with a private input set S_i, such that $|S_i| = k$, where $d > nk$ is the degree of an irreducible polynomial $f(t)$. Let $\mathcal{H}$ be a hash function from $\{0,1\}^*$ to $\{0,1\}^\ell$ where ℓ is defined by a security parameter.
Output: PPMU of $S_1, \cdots, S_n$

[Set up]
Each party P_i for $1 \leq i \leq n$:

1. computes $M_i'(t) = \big(t-(S_i)_1'\big) \cdots \big(t-(S_i)_k'\big)$ where $(S_i)_j' \leftarrow (S_i)_j || R_{ij}$ for random R_{ij} of which length is ℓ. If $M_i'(t)$ belongs to $\mathbb{G}_p$ then go to the next step; otherwise repeat step 1.
2. encrypts $M_i'(t)$ with random $r_i \in_R [0, 2^\ell]$ as

$$C_i(t) = (u_i(t), v_i(t)) = (g(t)^{r_i}, y(t)^{r_i} \cdot M_i'(t)) \bmod f(t).$$

[Step 1] Commitment
Each party P_i broadcasts h_i to all the parties where $h_i = \mathcal{H}(C_i(t)||POPK_i)$ for $POPK_i = ZK\{r_i | u_i(t) = g(t)^{r_i}\}$. (Note that we use non-interactive version for the proof of knowledge of the discrete logarithm.)

[Step 2] Broadcast Ciphertexts
Each party P_i collects all h_j's from other parties, and then broadcasts the ciphertext $C_i(t)$ to all other parties with $POPK_i = ZK\{r_i | u_i(t) = g(t)^{r_i}\}$.

[Step 3] Union
Each party P_i receives all $C_j(t)$'s and $POPK_j$'s from other parties and verifies if $h_j = \mathcal{H}(C_j(t)||POPK_j)$ for all j's. If verification fails then 'reject' the protocol. Otherwise, each party P_i multiplies all $C_j(t)$'s.

[Step 4] Decryption
All the parties perform (n, n)-threshold decryption to obtain the polynomial $\prod_{i=1}^n M_i'(t)$. If the form of the polynomial $\prod_{i=1}^n M_i'(t)$ is a product of nk linear factors, they 'accept' the result. Otherwise, 'reject'.

All the parties learn PPMU of $S_1, \cdots, S_n$ by performing polynomial factoring.

Fig. 2. Malicious Protocol for PPMU

In the malicious adversary model, an adversary can deviate from the protocol in an arbitrary fashion. Our honest-but-curious protocol does not prevent a malicious party from encrypting a fake polynomial after seeing other parties' polynomials. For example, consider the following situation. Let us assume that P_1 and P_2 broadcast their encryptions $C_1(t)$ and $C_2(t)$, respectively, and a malicious party P_3 prepares an arbitrary polynomial $C_3'(t)$. If the malicious P_3 broadcasts $C_3(t) := C_3'(t) \cdot \prod_{i=1}^{2} C_i(t)^{-1} \bmod f(t)$ as its encryption, after seeing $C_1(t)$ and $C_2(t)$, then P_1 and P_2 will get $C_3'(t)$ as the result of multiplication of $\prod_{i=1}^{3} C_i(t)$. Thus P_1 and P_2 receive a wrong result from the protocol. However, this guessing and removing attack can be blocked by randomizing the set input. In particular, we can avoid the attack by attaching an input element with a large random number of which length is in the security parameter. Thus, random padding in our protocol has two goals: making ElGamal semantically secure and avoiding the guessing and removing attack.

To extend our results to provide security against malicious adversaries, we also employ commitments and zero-knowledge proofs. The full decryption of the protocol is in Fig. 2.

Security Analysis We show that we can make a simulator $\mathbf{S}$ which translates any behavior of a malicious party A^* in the real model into the behavior of the party in the ideal model. Hence A^* in the real model gains no information other than what can be deduced in the ideal model. The following theorem is a formal statement of this property.

Theorem 4. *For any malicious party A^*, a simulator $\mathbf{S}$ in the ideal model exists in the random oracle model, such that the views of malicious party A^* and the honest parties in the real model is computationally indistinguishable from the views of the parties in the ideal model.*

Proof. Simulator $\mathbf{S}$ in the ideal model attempts to respond to malicious party's messages on behalf of honest parties, except that it never tell the inputs with which protocols are executed. For the simplicity of description we assume one malicious party A^* in this proof. Extension to multiple malicious parties is straightforward. The following sketch of how $\mathbf{S}$ operates suffices to show that the theorem holds.

1. For each simulated honest party P_i, $\mathbf{S}$ chooses a random commitment value $h_i \leftarrow \{0,1\}^\ell$ and performs step 1 of the protocol: $\mathbf{S}$ sends h_i to A^* and receives h^* from A^*. (The distribution of h_i is uniform in the range of $\mathcal{H}$.)

2. If there is no $\mathcal{H}$ query that outputs h^*, $\mathbf{S}$ stops. (In the random oracle model, the success probability of protocol is negligible without $\mathcal{H}$ query that outputs h^*.)

3. $\mathbf{S}$ extracts polynomial $M^*(t)$ from the hash inputs corresponding to h^*. Note that inputs contain the proof of plaintext knowledge. If inputs are not extractable, $\mathbf{S}$ stops. (In real world, if all the proofs of plaintext knowledge are not verified, the protocol fails.)

4. $\mathbf{S}$ obtains the roots of polynomial $M^*(t)$. If the number of roots is not k, $\mathbf{S}$ stops. (The malicious party tries to generate an encryption message on forged polynomials, for example, hoping to eliminate any linear factor in multiplication of linear factors in the real protocol. However, since the probability of the correct guessing of any linear factor is negligible and every party verifies the total number of elements after (n, n)-threshold decryption, the adversary is forced to generate encryption on a polynomial with k roots.)

5. $\mathbf{S}$ submits the set represented by these roots to the trusted third party. The honest party submits their private input sets to the trusted third party. The trusted third party returns the multiset union U to $\mathbf{S}$ and the honest players.

6. $\mathbf{S}$ chooses a set of polynomials $M_i'(t)$ such that $\prod_{i=1}^{n} M_i'(t) = U(t)/M^*(t)$ where $U(t)$ is the polynomial representation of U. $\mathbf{S}$ sets h_i as the hash query on inputs $(C_i'(t)||POPK_i)$, where $C_i'(t)$ is the encryption of $M_i'(t)$. (In this way, the second round message for each honest party is correctly formed so that the output of the protocol matches that of the trusted third party.)

7. **S** follows the rest of the protocol with A^* from step 2 and A^* learns the multiset union.

Note that the malicious party A^* cannot distinguish whether it interacts to **S** (in the ideal model) or it interacts to other honest parties (in the real world), and all parties learn the correct answer, in both the real and ideal models.

Extension to multiple malicious parties is simple: in the above step 4, **S** multiplies all polynomials and computes the roots of the polynomial. If the number of roots is not ck where c is the number of malicious parties, **S** stops.

5 Performance Analysis

In this section, we compare the performance of our PPMU protocol with that of *Kissner and Song (KS)'s Multiset Union* which uses their own 'Shuffling Protocol' in [13]. In KS's, to make an encryption of multiset union, the first party P_1 encrypts the k number of coefficients of his polynomial on $\mathbb{Z}_{N^2}$ and relays k ciphertexts to P_2. The other party $P_i (2 \leq i \leq n)$ performs simple exponentiations in turn to compute the product of an encrypted polynomial and his own unencrypted polynomial and relays ik ciphertexts.

Communication Round Regardless of whether the base network is a broadcast medium or a unicast medium, in the KS protocol, each party relays ciphertexts to the next party *in turn*. Thus, the n number of communication rounds are required. On the other hand, on the broadcast medium our protocol consists of *constant*-rounds because parties multiply all ciphertexts *in parallel*, without any relay.

Communication Cost In the KS protocol, since P_i relays the ik number of ciphertexts to P_{i+1}, the total number of bits needing to be transferred is $(k + 2k + 3k + \cdots + nk) \cdot \log N^2 \approx n^2 k \log N$. (Note that in Paillier cryptosystem ciphertexts are computed in $\bmod N^2$.) On the other hand, each party transmits only one ciphertext of size $\log q^{nk}$ in our protocol, and the total number of communication bits of our protocol is $n^2 k \log q$. Therefore, our protocol has the $\frac{\log q}{\log N}$ times smaller communication size than KS's. Recall that $\log q = \log |S| + \ell$ where S is the domain of sets and ℓ is defined by the security parameter. For example, $\log N$ is 1024 and ℓ is 160 in the 80-bit security level. If $|S| = 2^{30}$, then $\log q = 30 + 160 = 190$ and $\frac{\log q}{\log N} \approx 0.19$. That is, the communication size of our protocol is at least 5 times less than that in the KS's protocol, regardless of the communication medium, i.e., either broadcast or unicast.

Computation Cost The encryption time is similar for ours and KS's, but KS's is slightly better for large n and ours is better for large k. In KS's protocol, P_1 encrypts his own polynomial in $2k$ exponentiation on Z_{N^2}, and $P_i (2 \leq i \leq n)$ performs $((i-1)k+1)(k+1)$ exponentiation to compute the encryption of the product of polynomials. Therefore, the protocol totally requires $O(n^2 k^2)$ exponentiations. In our protocol, each party encrypts his own polynomial in two exponentiations in $\mathbb{F}_{q^d}$. It totally involves only $O(n)$ exponentiations. However, our protocol performs operations over $\mathbb{F}_{q^d}$, hence two exponentiations involve a constant number of multiplications over $\mathbb{F}_{q^d}$, which take $O(d^{\log_2 3})$ using Karatsuba method or $O(d \log d \log \log d)$ by fast Fourier transform (FFT) [22]. Therefore, our protocol totally requires $O(n^{1+\log_2 3} k^{\log_2 3})$ using Karatsuba method.

Polynomial Factoring A polynomial factoring algorithm is required in both KS's and our PPMU protocol at the last step. KS utilize an additively homomorphic cryptosystem which supports an efficient polynomial factoring algorithm. However, such a polynomial factoring algorithm is not known yet. On the other hand, in our protocol, the set union, i.e., the result of the PPMU protocol, is represented by an element in a finite field of known order, and there exist efficient factoring algorithms running in the field. Using a square-free decomposition and the Cantor-Zassenhaus algorithm [22], a polynomial of degree nk is factored in $O((nk)^{2+o(1)} \cdot \log q)$ operations in $\mathbb{F}_q$.

6 Conclusion

In this paper, we propose constant-round protocols for privacy preserving multiset union using ElGamal on polynomials. Our protocol improves communication overhead in comparison to the previous protocol. We prove the security of our protocols in the random oracle model.

References

1. F. Boudot, B. Schoenmakers, and J. Traoré. A fair and efficient solution to the socialist millionaires' problem. *Discrete Applied Mathematics*, 111(1-2):23–36, 2001.
2. J. Brickell and V. Shmatikov. Privacy-preserving graph algorithms in the semi-honest model. In *ASIACRYPT*, pages 236–252, 2005.
3. M. Chateauneuf, A. C. H. Ling, and D. R. Stinson. Slope packings and coverings, and generic algorithms for the discrete logarithm problem. *Journal of Combinatorial Designs*, 11(1):36–50, 2003.
4. Y. Desmedt and Y. Frankel. Threshold cryptosystems. In *CRYPTO*, pages 307–315, 1989.
5. A. V. Evfimievski, J. Gehrke, and R. Srikant. Limiting privacy breaches in privacy preserving data mining. In *PODS*, pages 211–222, 2003.
6. M. J. Freedman, K. Nissim, and B. Pinkas. Efficient private matching and set intersection. In *EUROCRYPT*, pages 1–19, 2004.
7. K. B. Frikken. Privacy-preserving set union. In *ACNS*, pages 237–252, 2007.
8. J. Furukawa and K. Sako. An efficient scheme for proving a shuffle. In *CRYPTO*, pages 368–387, 2001.
9. O. Goldreich. *Foundations of Cryptography*, volume 2. Cambridge University Press, 2004.
10. B. A. Huberman, M. K. Franklin, and T. Hogg. Enhancing privacy and trust in electronic communities. In *ACM Conference on Electronic Commerce*, pages 78–86, 1999.
11. M. Kantarcioglu and C. Clifton. Privacy-preserving distributed mining of association rules on horizontally partitioned data. *IEEE Trans. Knowl. Data Eng.*, 16(9):1026–1037, 2004.
12. L. Kissner and D. Song. Privacy-preserving set operations. In *Advances in Cryptology – CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 241–257. Springer-Verlag, 2005.
13. L. Kissner and D. Song. Private and threshold set-intersection. Technical Report CMU-CS-05-113, Carnegie Mellon University, February 2005.
14. H. Lipmaa. Verifiable homomorphic oblivious transfer and private equality test. In *ASIACRYPT*, pages 416–433, 2003.
15. C. A. Neff. A verifiable secret shuffle and its application to e-voting. In *ACM Conference on Computer and Communications Security*, pages 116–125, 2001.
16. C. M. O'Keefe, M. Yung, L. Gu, and R. A. Baxter. Privacy-preserving data linkage protocols. In *WPES*, pages 94–102, 2004.
17. T. P. Pedersen. A threshold cryptosystem without a trusted party (extended abstract). In *EUROCRYPT*, pages 522–526, 1991.
18. K. Peng, C. Boyd, E. Dawson, and B. Lee. An efficient and verifiable solution to the millionaire problem. In *ICISC*, pages 51–66, 2004.
19. J. M. Pollard. Monte carlo methods for index computation $\mod p$. *Mathematics of Computation*, 32.
20. Y. Sang and H. Shen. Privacy preserving set intersection based on bilinear groups. In *ACSC*, pages 47–54, 2008.
21. V. Shoup. Lower bounds for discrete logarithms and related problems. In *EUROCRYPT*, pages 256–266, 1997.
22. V. Shoup. *A computational introduction to Number Theory and Algebra.* Cambridge University Press, 2005.
23. A. C. Yao. Protocols for secure computations. In *Foundations of Computer Science, 1982. SFCS '08. 23rd Annual Symposium on*, pages 160–164, 1982.