Privacy Preserving Multiset Union with ElGamal Encryption

Jeongdae Hong[1], Jung Woo Kim[1], and Jihye Kim[2] and Kunsoo Park[1], and Jung Hee Cheon[3]

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

Abstract. The 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 an efficient PPMU protocol, using multiplicative homomorphic property of ElGamal encryption over $\mathbb{F}_q[x]/f(x)$ where q is a prime and $f(x)$ is an irreducible polynomial over $\mathbb{F}_q$. The protocol involves a constant number of rounds and improves the computation and communication complexities of the scheme proposed by Kissner and Song. We also prove the security of the protocol in the random oracle model.

Key words: Privacy, Multiset Union, ElGamal Encryption, Homomorphic Encryption

1 Introduction

Privacy preserving set operations have been actively studied and examples include set intersection [8, 15, 6, 11] and set union [14, 12, 3]. One of operations, important but comparatively paid less attention, is *multiset union*.

Privacy preserving multiset 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 a 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. PPMU ensures that a set of parties collaboratively compute multiset union *secretly*, meaning that no party learns more information about other parties' private inputs than what can be deduced from the result of union.

The literature of multiset union includes the scheme proposed by Kissner and Song (KS) [14]. Protocols in [14] use additive homomorphic encryption proposed by Paillier [16] for privacy preserving. Each party in protocols should wait for other parties' intermediate multiplications, since the Paillier cryptosystem does not support the product of two encrypted polynomials but the product of an unencrypted polynomial and an encrypted polynomial. In other words, if multi-parties want the product of encrypted polynomials, they should compute the product in turn. Thus, the number of rounds (i.e., delay) increases linearly in the number of participants. In addition, the threshold version of Paillier [7] needs a fully trusted dealer for distributing private key shares; however, the trusted third party (TTP) in general is undesirable because all of the parties should totally trust it. These disadvantages of KS motivate us to construct efficient multiset union protocols without TTP. Moreover, considering that multiset union is a simpler operation than set union, it is natural to think about light-weight protocols, solely optimized for multiset union.

In this paper, we propose efficient privacy-preserving multiset union (PPMU) protocols using ElGamal cryptosystem and its multiplicative homomorphic property [5]. In our protocol, the plaintext polynomial is selected in a multiplicative cyclic group $(\mathbb{F}_q[x]/f(x))^*$ where q is a prime number and $f(x)$ is an irreducible polynomial of degree d. The ElGamal encryption and decryption are implemented in $(\mathbb{F}_q[x]/f(x))^*$, which is isomorphic to the finite field $\mathbb{F}_{q^d}$. We call it *ElGamal with polynomials*. Using multiplicative homomorphic property of ElGamal with polynomials, every party in the protocol

can compute the product of encrypted polynomials simultaneously. Hence the proposed scheme requires only a constant round of communications, and improves the computation and communication complexities of the Kissner and Song's scheme [14] based on Paillier cryptosystem. Our implementation shows that the proposed scheme is about 30 times faster than the previous one. Moreover, the proposed PPMU is easily combined with the threshold decryption of ElGamal [4, 9] which does not require TTP.

2 Privacy Preserving Multiset Union (PPMU) with ElGamal

2.1 Problem Setting and Polynomial Representation of a Set

Problem Setting In this paper, we follow KS's definition and technique for PPMU. Let f and g be polynomial representations of multisets S and T. They defined the union $S \cup T$ as a multiset, where each element a that appears $b_S \geq 0$ times in S and $b_T \geq 0$ times in T, appears in the resulting multiset $b_S + b_T$ times. Then, one computes the polynomial representation of $S \cup T$ as $f * g$.

Polynomial Representation of a Set by an Element in $\mathbb{F}_{q^d}$ We represent a set of elements by a polynomial in $(\mathbb{F}_q[x]/f(x))^*$, which is isomorphic to the finite field $\mathbb{F}_{q^d}$, where q is a prime number and $f(x)$ is an irreducible polynomial of degree d. For example, given a set of k elements $S_i = \{m_1, m_2, \cdots, m_{d_k}\}$ where each m_i is an element of $\mathbb{F}_q$, we construct its polynomial representation as $m(x) = \prod_{i=1}^{k}(x - m_i)$. We can easily implement ElGamal cryptosystem with polynomials in $(\mathbb{F}_q[x]/f(x))^*$ and extend it to a threshold version.

2.2 Privacy Preserving Multiset Union (PPMU) Protocol

We consider both honest-but-curious and malicious models by Goldreich [10]. For the formal definitions of these models, refer to [14]. First, we propose a protocol for the honest-but-curious model. Next, we consider several possible attacks and its counter-measures; then, we extend our honest-but-curious protocol to the malicious model.

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 $|S_i| = k$. Let $(S_i)_j$ be the j-th element of set S_i. Our protocol allows all the parties to learn the union $S_1 \cup S_2 \cup \cdots \cup S_n$ with multiplicity, but no party can gain a non-negligible advantage in learning which set S_i an element is in. The secret key a corresponding to the public key is shared to all the parties. Note that this scheme can be easily combined with a threshold ElGamal cryptosystem to provide a PPMU scheme without TTP [17].

Define a key set

$$\mathcal{K} = \{(f(x), \alpha(x), a, \beta(x)) : \beta(x) \equiv \alpha(x)^a \bmod f(x)\} \tag{2.1}$$

where $\alpha(x), \beta(x), f(x)$ are public keys and a is a private key.

In encryption phase, each party P_i calculates a polynomial $m_i(x) = (x - (S_i)_1) \cdots (x - (S_i)_k)$ from his set S_i, encrypts it with a random r_i as

$$c_i(x) = (\alpha(x)^{r_i},\ \beta(x)^{r_i} \cdot m_i(x)) \bmod f(x) \tag{2.2}$$

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

$$C(x) = (C_1(x), C_2(x)) = (\alpha(x)^{r_1+\cdots+r_n},\ \beta(x)^{r_1+\cdots+r_n} \cdot \prod_{i=1}^{n} m_i(x)) \tag{2.3}$$

In decryption phase, for a given ciphertext $C(x)$ all the parties perform a group decryption [4] to obtain the polynomial $\prod_{i=1}^{n} m_i(x)$. Then, all the parties learn all of the elements in the polynomial by using polynomial factoring [18].

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 with polynomials is as secure as the usual ElGamal scheme defined over a prime field.[1]

Protocol for the Malicious (MAL) Model In the malicious adversary model, an adversary will deviate from the protocol in an arbitrary fashion. In our honest-but-curious protocol the following attacks are possible.

- *Guess & Elimination.* An adversary can change the result of the protocol. Especially, the number of elements of the PPMU can be reduced by the adversary. Any inverse polynomial of some valid polynomial can eliminate valid elements of other parties. Let us consider the following example. Assume that $n = 3$ and P_3 is an adversary. Let $m_1(x) = (x-a)(x-b)$, $m_2(x) = (x-a)(x-c)$, and $m_3(x) = (x-a)(x-b)^2(x-c)^{-1}$. Then the result of the protocol will be $(x-a)^3(x-b)^3$. That is, the element c is eliminated by an adversary.
- *Delayed Participation.* The inverse of any ciphertext can eliminate a valid ciphertext. If P_3 publishes $c_3(x) := c'_3(x) \cdot \prod_{i=1}^{2} c_i(x)^{-1}$ on the bulletin board, P_1 and P_2 will get $c'_3(x)$ as the result of multiplication of $\prod_{i=1}^{3} c_i(x)$. Thus P_1 and P_2 not only receive a wrong result, but also even do not know the fact that they receive a wrong one.

To prevent the *Guess & Elimination* attack, we append a random number to every element of sets: $a' \leftarrow a||R_i$, where $a||b$ denotes a concatenated with b and R_i denotes a random number. In the previous example, the probability that an adversary can make the same form of elements is $1/2^w$ where w is the number of bits of random numbers.

To prevent the *Delayed Participation* attack, we utilize commitment and a zero-knowledge proof of plaintext knowledge (POPK). Commitment prevents delayed participants from learning any information about the message and POPK forces participants to generate messages in a correct form, e.g., encryption of k-root polynomial in our case.

The malicious protocol has several differences from the honest-but-curious one, and the counter measures mentioned above play an important role in protecting our protocol from adversaries. The full description of the protocol is in Fig. 1.

Security Analysis We show that we can make a simulator **S** which translates any behavior of the 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 more information than what can be deduced in the ideal model. The following lemma is a formal statement of this property.

Theorem 1. *For any malicious party A^*, simulator* **S** *in the ideal model exists in the random oracle model, such that the multiset union of the malicious party A^* and the honest parties in the real model is computationally indistinguishable from the multiset union of the parties in the ideal model.*

Proof. (**Sketch)** Simulator **S** in the ideal model attempts to respond to malicious party's messages on behalf of honest parties, except that it is never told the inputs with which protocols are executed. The high level view of the proof is the same as the one in Figure 12 in section C.2 in [13]. For the simplicity of description we assume one malicious party A^* in this proof. We can extend the proof to multiple malicious parties. The sketch of how **S** operates is as follows:

1. For each simulated honest party i, **S** chooses random commitment value $h_i \leftarrow \{0,1\}^\ell$ and performs step 1 of the protocol: **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^*, **S** stops. (The probability of protocol success without such $\mathcal{H}$ query that outputs h^* is negligible in the random oracle model.)

[1] Let (c_1, c_2) be a ciphertext of a message $m(x) = \prod_{i=1}^{k}(x - m_i)$ where $c_1 = (\alpha(x)^r \bmod f(x))$ and $c_2(x) = (\beta(x)^r m(x) \bmod f(x))$. One may think that finding a factor $(x - m_i)$ from the ciphertext would be easy. However, $c_2(x)$ is of the form $\beta(x)^r m(x) - g(x)f(x)$ for some $g(x) \in \mathbb{F}_q[x]$ and so $c_2(m_i) = 0$ if and only if $g(m_i) = 0$, which happens with probability $1/q$. This makes that $m_i \in S$ is indistinguishable from $m' \notin S$.

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 $f(x)$. Let $\mathcal{H}$ be a hash function from $\{0,1\}^*$ to $\{0,1\}^\ell$ where ℓ is a security parameter.
Output: PPMU of $S_1, \cdots, S_n$

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

1. appends a random number R_{ij} to $(S_i)_j$ such as $(S_i)'_j \leftarrow (S_i)_j || R_{ij}$.
2. calculates the polynomial $m_i(x) = \big(x - (S_i)'_1\big) \cdots \big(x - (S_i)'_k\big)$.
3. for a given key as (2.1), encrypts the polynomial $m_i(x)$ with a random r_i and s_i as equation (2.2).

[Step 1] Commitment
Each party P_i sends h_i to all the parties where $h_i \leftarrow \mathcal{H}(c_i(x)||POPK_i)$.

[Step 2] Broadcast
Each party P_i sends the encryption $c_i(x)$ to all the parties with $POPK_i$.

[Step 3] Union
For $1 \le j \le n$, each party P_i

1. verifies the existence $h_j \leftarrow \mathcal{H}(c_j(x)||POPK_j)$ from $c_j(x)$ and $POPK_j$.
2. multiplies all the $c_j(x)$ as equation (2.3).

[Step 4] Decryption
All the parties perform a group decryption to obtain the polynomial $\prod_{i=1}^{n} m_i(x)$. If the form of the polynomial $\prod_{i=1}^{n} m_i(x)$ 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. 1. Malicious Protocol for PPMU

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

4. **S** obtains the roots of polynomial $m^*(x)$. If the number of roots is not k, **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 group decryption, the adversary is forced to generate encryption on a polynomial with k roots.)

5. **S** submits the set represented by these roots to the trusted third party. The honest players submit their private input sets to the trusted third party. The trusted third party returns the multiset union U to **S** and the honest players.

6. **S** chooses a set of polynomials $m_i(x)$ s.t. $\prod_{i=1}^{n} m_i(x) = U(x)/m^*(x)$ where $U(x)$ is the polynomial representation of U. **S** sets as h_i the hash query on inputs $(c_i(x)||POPK_i)$. (In this way, the second round message for each honest party is correctly formed so that the protocol output matches the third party output.)

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 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 step 4, **S** multiplies all

polynomials and computes the roots of polynomials multiplication. If the number of roots is not ck where c is the number of malicious parties, **S** stops.

3 Discussion

3.1 Performance Analysis and Comparison

We compare the performance of our PPMU protocol with that of *KS's Multiset Union* which is composed of the first 4 steps of their 'Over-Threshold Set-Union Protocol' in [14] and group decryption. Table 1 compares the performances of two protocols in communication and computation cost. A remarkable feature of our protocol is that it requires only constant rounds for communication while KS protocol requires $O(n)$ rounds. The encryption time is similar for ours and KS protocol, but KS is slightly better for large n and ours is better for large k. However, we remark that for small nk our protocol is more efficient than KS protocol since one ElGamal encryption covers many elements of a multiset, which will be demonstrated in the next subsection.

		# of Rounds	Communication	Enc.	Dec.
HBC	KS [14]	n	$n\mathcal{U}$ of size $O(nk)$	$O(n^2k^2)$	$O(nk)$
	Ours	1	$n\mathcal{B}$ of size $O(nk)$	$O(n)\cdot\mathbf{MUL}(d)$	$O(1)\cdot\mathbf{MUL}(d)$
MAL	KS [14]	$n+1$	$n\mathcal{B}$ of size $O(nk)$ and $n\mathcal{B}$ of size $O(k)$	$O(n^2k^2)$	$O(nk)$
	Ours	2	$n\mathcal{B}$ of size $O(nk)$	$O(n)\cdot\mathbf{MUL}(d)$	$O(1)\cdot\mathbf{MUL}(d)$

$n\mathcal{B}$ and $n\mathcal{U}$ denote n Broadcasts and n Unicasts. $\mathbf{MUL}(d)$ represents the complexity for a multiplication of polynomials of length d over $\mathbb{F}_q$, in which $d > nk$ and $\mathbf{MUL}(d) = O(d^{\log_2 3})$ or $O(d\log d\log\log d)$.

Table 1. Performance Comparison Our Protocol with KS's Multiset Union

In HBC protocol of KS's, to make an encryption of multiset union, the first party P_1 encrypts his polynomial in $2k$ exponentiations on $\mathbb{Z}_N$ and relays $O(k)$ ciphertexts. The other party P_i $(2 \le i \le n)$ performs simple exponentiations in turn to compute the product of an encrypted polynomial and his own unencrypted polynomial with $((i-1)k+1)(k+1)$ exponentiations and relays ik ciphertexts; therefore, the protocol totally requires $O(n^2k^2)$ exponentiations. In our protocol, each party simultaneously encrypts his own message in two exponentiations in $\mathbb{F}_{q^d}$ and broadcasts the ciphertext. Then, each party computes the product of the n ciphertexts. It involves only $O(1)$ ciphertexts and $O(n)$ exponentiation. However, our protocol performs operations over $\mathbb{F}_{q^d}$ and d must be larger than nk. Hence the size of the ciphertext becomes $O(nk)$ and two exponentiations, which involve a constant number of multiplications over $\mathbb{F}_{q^d}$, take $O(d^{\log_2 3})$ using Karatsuba method or $O(d\log d\log\log d)$ by fast Fourier transform (FFT) [18, p.415]. We remark that faster multiplication methods such as Karatsuba or FFT cannot be applied to KS protocol since each exponentiation is performed independently.

3.2 Experimental Results

We present experimental results to verify the performance gains of our protocol. Our experiments were conducted on a 3.20-GHz Intel Pentium IV CPU with 1GB memory, Red Hat 8, gcc 4.3.0 and NTL [1] which is a high-performance library for doing number theoretic computation. For practically secure use of the Paillier cryptosystem, $\log N \ge 1024$ is recommended. Thus, we implemented both protocols for the parameters, $k = 10$, $n = 10$, $\log p = 160$ and $\log q^d = \log N = 1024$. The computation time of our protocol is 22.28 seconds, which is about 33 times faster than 703.54 seconds of KS's protocol. In addition, if we account for delivery time, the performance gain will be more prominent.

Although we did not perform the experiment in Optimal Extension Field (OEF), it is known that OEF is the best choice [19] for the software implementation. We recommend that ElGamal cryptosystem with polynomials is implemented over OEF [2].

	Enc.	Dec.	Total
KS	701.40 sec	2.14 sec	703.54 sec
Ours	15.03 sec	7.25 sec	22.28 sec

Table 2. Comparison of Experimental Results

4 Conclusion & Future Work

In this paper, we proposed efficient privacy preserving multiset union protocols. It requires only constant-round communication and about 30 times faster encryption than the previous one. However, there still remains much room for developing various practical applications, since we have not found solutions for other set operations such as set intersection, element reduction and set union. Another issue is protocol robustness. Although the current version of protocol is secure, its performance degrades whenever any fault happens. For robustness, the protocol is required to detect malicious players. A new cryptographic tools such as Zero-knowledge Proof of the Degree of a Message must be studied.

References

1. www.shoup.net/ntl/.
2. D. V. Bailey and C. Paar. Optimal extension fields for fast arithmetic in public-key algorithms. In *CRYPTO*, pages 472–485, 1998.
3. J. Brickell and V. Shmatikov. Privacy-preserving graph algorithms in the semi-honest model. In *ASIACRYPT*, pages 236–252, 2005.
4. Y. Desmedt and Y. Frankel. Threshold cryptosystems. In *CRYPTO*, pages 307–315, 1989.
5. T. ElGamal. A public key cryptosystem and signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, JULY 1985.
6. A. V. Evfimievski, J. Gehrke, and R. Srikant. Limiting privacy breaches in privacy preserving data mining. In *PODS*, pages 211–222, 2003.
7. P.-A. Fouque, G. Poupard, and J. Stern. Sharing decryption in the context of voting or lotteries. In *Financial Cryptography*, pages 90–104, 2000.
8. M. J. Freedman, K. Nissim, and B. Pinkas. Efficient private matching and set intersection. In *EUROCRYPT*, pages 1–19, 2004.
9. R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Secure distributed key generation for discrete-log based cryptosystems. *J. Cryptology*, 20(1):51–83, 2007.
10. O. Goldreich. *Foundations of Cryptography*, volume 2. Cambridge University Press, 2004.
11. 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.
12. 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.
13. L. Kissner and D. Song. *Private and threshold set-intersection.* Carnegie Mellon University. Technical Report CMU-CS-05-113.
14. 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.
15. C. M. O'Keefe, M. Yung, L. Gu, and R. A. Baxter. Privacy-preserving data linkage protocols. In *WPES*, pages 94–102, 2004.
16. P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *EUROCRYPT*, pages 223–238, 1999.
17. T. P. Pedersen. A threshold cryptosystem without a trusted party (extended abstract). In *EUROCRYPT*, pages 522–526, 1991.
18. V. Shoup. *A computational introduction to Number Theory and Algebra.* Cambridge University Press, 2005.
19. N. P. Smart. A comparison of different finite fields for use in elliptic curve cryptosystems. In *Computers and Mathematics with Applications*, pages 91–100, 2001.