# Gmatch: Secure and Privacy-Preserving Group Matching in Social Networks

Boyang Wang †,††, Baochun Li †† and Hui Li †
† State Key Laboratory of Integrated Services Networks, Xidian University, Xi'an, China
†† Department of Electrical and Computer Engineering, University of Toronto, Toronto, Ontario, Canada
{bywang, lihui}@mail.xidian.edu.cn, bli@eecg.toronto.edu

***Abstract*—Groups are becoming one of the most compelling features in both online social networks and Twitter-like micro-blogging services. A stranger outside of an existing group may have the need to find out more information about attributes of current members in the group, in order to make a decision to join. However, in many cases, attributes of both group members and the stranger need to be kept private and should not be revealed to others, as they may contain sensitive and personal information. How can we find out matching information exists between the stranger and members of the group, based on their attributes that are not to be disclosed? In this paper, we present a new group matching mechanism, by taking advantage private set intersection and ring signatures. With our scheme, a stranger is able to collect correct group matching information while sensitive information of the stranger and group members are not disclosed. Finally, we propose to use batch verification to significantly improve the performance of the matching process.**

## I. Introduction

As online social networks and Twitter-like micro-blogging services redefine our lifestyle, groups are becoming one of the most frequently used features. Groups are, in general, formed with common attributes, such as geographic locations and hobbies. However, the features of a group are generally described by only a few keywords or a short description, which sometimes is not enough for users to make decisions when choosing an appropriate group for themselves. Especially, when several groups have similar (or even the same) keywords and descriptions, it is very inconvenient for users to choose the most suitable one among these groups. In order to make a better decision when choosing a group to join, a stranger with a profile of his own attributes — who is still an outsider of the group — needs to collect detail matching information from all the group members' profiles. Such a problem is referred as to *group matching*.

In most situations, attributes of users are sensitive, such as personal health records and religious preferences. It is typical for a user to store these attributes privately [1], so that only his friends or members in the same group are able to reveal these attributes, but strangers or any third party cannot learn these sensitive information. Unfortunately, collecting group matching information using these sensitive attributes may introduce a number of privacy problems. On one hand, since the stranger is not familiar with the group, the stranger does not want to reveal his sensitive attributes to any group member during the matching process. On the other hand, because the stranger is an outside and untrusted user to the group, each group member is reluctant to reveal his own attributes and the exact matching results between two entities to the stranger.

To make matters more challenging, each group member needs to generate a signature on his matching response, which contains matching information between the stranger and himself, and sends the signature and the matching response together to the stranger, so that the stranger is convinced the matching response is reliable and correct. Unfortunately, due to the *unforgeability* of signatures (only the entity with the knowledge of the private key can create valid signatures), the stranger is able to learn the identity of the signer on each matching response, and reveal exact matching information between himself and each group member.

In this paper, we proposed Gmatch, a novel secure and privacy-preserving group matching scheme in online social networks. We utilize private set intersection [2] in Gmatch, so that the stranger is able to collect matching information from the group while both the stranger and each group member are able to preserve sensitive attributes to each other. Meanwhile, with ring signatures [3], [4], the stranger is convinced that matching information from the group is correct, but he cannot learn exact matching information between himself and each group member. In addition, we improve the efficiency of the matching process using batch verification.

The remainder of this paper is organized as follows: In Section II, we introduce the system model and design objectives. In Section III, we briefly describe cryptographic primitives we utilized in Gmatch. We then present the details of Gmatch in Section IV. Section V provides a thorough security analysis, and Section VI evaluates the performance of Gmatch. Finally, we briefly discuss related work in Section VII, and conclude this paper in Section VIII.

## II. Problem Statement

### A. System Model

Our system is a social network, which includes a *stranger* $S$ and all $d$ group members $P_1$, ..., $P_d$ in the group $\mathcal{P}$ (as shown in Fig. 1). The stranger $S$, who is not a member of the group $\mathcal{P}$, has $k$ attributes in his profile and the $j$-th attribute is denoted as $a_{s,j}$. The stranger's profile is denoted as $\mathcal{A}_s = \{a_{s,1}, ..., a_{s,k}\}$. Group member $P_i$ has $m$ attributes and the profile of this group member is denoted as $\mathcal{A}_i = \{a_{i,1}, ..., a_{i,m}\}$. In our model, we assume all group members have the same size of profile. Attributes in every user's profile are private and

sensitive, which are stored and maintained locally by each user. Note that we also assume there does not exist of a third party that first collects all the group members' profiles, and then simply completes group matching between itself and the stranger. Even if there exists a group manager who maintains basic activities of the group, such as the changes of membership, it is still not able to access sensitive attributes of group members. The stranger completes group matching in a *distributed* manner [5].

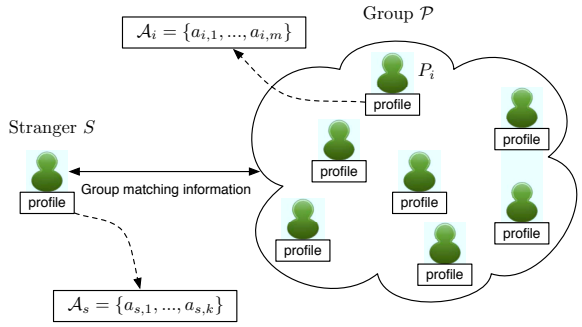



Fig. 1. Stranger $S$ wants to collect group matching information from group $\mathcal{P}$ based on his attribute set $\mathcal{A}_s$.

During group matching, this stranger $S$ wishes to collect group matching information from group $\mathcal{P}$ based on his profile. If an attribute in a group member's profile is equal to an attribute in the stranger's profile, it is then referred to as a *matched attribute*. Otherwise, it is called an *unmatched attribute*. The total number of group members that has the same attribute with the attribute $a_{s,j}$, is denoted as the *matching degree* $D_j$ of attribute $a_{s,j}$. The group matching information from the group $\mathcal{P}$ is described as $\mathcal{D}(\mathcal{P}) = \{D_1, ..., D_k\}$. Each group member $P_i$ is asked to provide matching information to stranger $S$ based on profile $\mathcal{A}_i$, so stranger $S$ can calculate group matching information $\mathcal{D}(\mathcal{P})$ from group $\mathcal{P}$.

### B. Privacy Threats

In this paper, we assume the stranger is honest-but-curious. It means the stranger will honestly follow the protocol to collect group matching information, but may attempt to learn more information than allowed.

### C. Design Objectives

During the group matching, our scheme should be able to provide the following desirable privacy properties. (1) **Stranger's Attributes Privacy**: The stranger does not reveal any attribute in his profile to any group member. (2) **Group Members' Attributes Privacy**: The stranger only obtains matched attributes that both in his profile and some group member's profile, while the unmatched attributes in group members' profiles are not disclosed to the stranger. (3) **Exact Matching Information Privacy**: The stranger is able to compute group matching information, while any exact matching information between himself and each group member is not revealed.

## III. Preliminaries

In this section, we briefly introduce cryptographic primitives that we implement in Gmatch.

### A. Bilinear Maps

Let $G_1$, $G_2$ and $G_T$ be three multiplicative cyclic groups of prime order $p$, $g_1$ be a generator of $G_1$, and $g_2$ be a generator of $G_2$. A bilinear map $e$ is a map $G_1 \times G_2 \rightarrow G_T$ with the following properties: (1) **Computability**: there exists an efficient algorithm for computing map $e$. (2) **Bilinearity**: for all $u \in G_1$, $v \in G_2$ and $a, b \in Z_p$, $e(u^a, v^b) = e(u, v)^{ab}$. (3) **Non-degeneracy**: $e(g_1, g_2) \neq 1$.

### B. Ring Signatures

The concept of ring signatures was first proposed by Rivest *et al.* in 2001 [3]. A ring signature scheme has the property that, a verifier is convinced that a ring signature was produced using one of group members' private keys, but this verifier is not able to determine which one.

### C. Private Set Intersection

Private set intersection [2], [6], [7] enables two parties to calculate the intersection of their private sets without leaking any additional information. Private set intersection can be construct using additive homomorphic encryption, such as Paillier cryptosystem [8]. The additive homomorphic encryption algorithm $\mathsf{Enc}(\cdot)$ in [8] is able to complete following operations, without knowing the corresponding plaintexts.

- Given $\mathsf{Enc}(m_1)$ and $\mathsf{Enc}(m_2)$, output $\mathsf{Enc}(m_1 + m_2) = \mathsf{Enc}(m_1) \cdot \mathsf{Enc}(m_2)$.
- Given $\mathsf{Enc}(m_1)$ and a constant $c$, output $\mathsf{Enc}(c \cdot m_1) = \mathsf{Enc}(m_1)^c$.

## IV. Gmatch: Secure and Privacy-Preserving Group Matching

### A. Overview

In this section, we introduce Gmatch, a secure and privacy-preserving group matching scheme. By utilizing private set intersection, the stranger can learn the matching information from the group without revealing any unmatched attributes in group members' profiles. With ring signatures, the stranger is convinced that a matching response is correct and generated by a group member, yet cannot distinguish this matching response belongs to which particular group member. Exploiting the properties of bilinear maps, Gmatch can support batch verification, which is able to greatly improve the efficiency of verification of ring signatures. In addition, with minor modifications in the construction of Gmatch, we can achieve even higher privacy levels.

### B. Gmatch

Gmatch includes four steps: ***Setup***, ***Compute***, ***Evaluate***, ***Match***. In ***Setup***, stranger $S$ and each group member generate their own public/private key pairs. In ***Compute***, stranger $S$ first generates a polynomial, where each attribute in his profile is a root of this polynomial and all the roots are in his profile.

Then, stranger $S$ encrypts all the coefficients of this polynomial by performing additive homomorphic encryption, and sends all the encrypted coefficients to all the group members. In ***Evaluate***, each group member evaluates a matching value for each attribute in his own profile using all the encrypted coefficients, signs a matching response that contains all the matching values generated by himself, and sends this matching response and the corresponding signature to the stranger. In ***Match***, stranger $S$ first checks the correctness of a matching response by verifying its signature, then computes whether each matching value in this matching response indicates a matched attribute. After collecting all the matching responses from all group members, the stranger $S$ calculates matching degrees for all the attributes in his profile. Details of each step are listed as follows.

***Setup.*** Stranger $S$ generates his public/private key pair $(pk_s, sk_s)$ for additive homomorphic encryption. Here, we utilize Paillier cryptosystem [8]. The encryption algorithm is denoted as $\mathsf{Enc}$, and the corresponding decryption algorithm is denoted as $\mathsf{Dec}$. Each group member generate his public/private key pair $(pk_i, sk_i)$ for computing ring signatures. The ring signature scheme we used is BGLS [4], which is based on bilinear maps. The total number of group members is $d$. The number of attributes in the stranger's profile is $k$, and the number of attributes in each group member's profile is $m$.

**Algorithm 1** $\mathsf{KeyGen}$

Given two multiplicative cyclic groups $G_1$, $G_2$ with prime order $p$ and their generators $g_1$, $g_2$ respectively, group member $P_i$ generates his public key and private key as:

1) Pick random $u_i \in Z_p$.
2) Compute $v_i = g_2^{u_i} \in G_2$.

Group member $P_i$'s public key is $pk_i = v_i$ and his private key is $sk_i = u_i$.

***Compute.*** Stranger $S$ first constructs a $k$-degree polynomial $P(x)$, whose $k$ roots are all attributes in his profile. This polynomial is described as:

$$P(x) = (x - a_{s,1})(x - a_{s,2}) \dots (x - a_{s,k}) = \sum_{i=0}^{k} \alpha_i x^i. \quad (1)$$

Clearly, if an attribute $a_{i,j}$ from group member $P_i$ is a matched attribute that equals some attribute in stranger $S$'s profile, then $a_{i,j}$ is also a root of this $k$-degree polynomial $P(x)$, and we have $P(a_{i,j}) = 0$.

After generating polynomial $P(x)$, stranger $S$ encrypts all the $k+1$ coefficients of this polynomial $P(x)$ using $\mathsf{Enc}$ with his public key $pk_s$. He then sends all the $k+1$ encrypted coefficients $\{\mathsf{Enc}(\alpha_0), ..., \mathsf{Enc}(\alpha_k)\}$ to each group member (as illustrated in Fig. 2).

***Evaluate.*** Group member $P_i$ has $m$ attributes and evaluates a matching value $w_{i,j}$ for each attribute $a_{i,j}$ in his profile. More specifically, group member $P_i$ first computes an encrypted polynomial value $\mathsf{Enc}(P(a_{i,j}))$ for each attribute

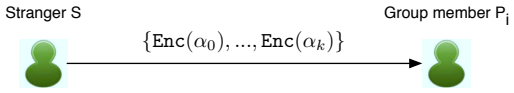



Fig. 2. Stranger $S$ sends all the encrypted coefficients to group member $P_i$.

$a_{i,j}$. Due to properties of additive homomorphic encryption we introduced in Section III, this encrypted polynomial value $\mathsf{Enc}(P(a_{i,j}))$ can be easily computed by $P_i$'s attribute $a_{i,j}$ and all the encrypted coefficients $\mathsf{Enc}(\alpha_i)$, for $i \in [0, k]$, as follows:

$$\begin{aligned} & \mathsf{Enc}(P(a_{i,j})) \\ = \ & \mathsf{Enc}(\alpha_0 + \alpha_1 a_{i,j} + \cdots + \alpha_k a_{i,j}^k) \\ = \ & \mathsf{Enc}(\alpha_0) \times \mathsf{Enc}(\alpha_1)^{a_{i,j}} \times \cdots \times \mathsf{Enc}(\alpha_k)^{a_{i,j}^k}. \end{aligned} \quad (2)$$

After that, group member $P_i$ generates a random number $\tau_{i,j}$, and computes a matching value $w_{i,j}$ of attribute $a_{i,j}$ as:

$$\begin{aligned} w_{i,j} &= \mathsf{Enc}(\tau_{i,j} \cdot P(a_{i,j}) + a_{i,j}) \\ &= \mathsf{Enc}(P(a_{i,j}))^{\tau_{i,j}} \times \mathsf{Enc}(a_{i,j}), \end{aligned} \quad (3)$$

where $\mathsf{Enc}(a_{i,j})$ can be computed using the stranger's public key $pk_s$ and attribute $a_{i,j}$ with $\mathsf{Enc}$.

Then, group member $P_i$ constructs his matching response $\boldsymbol{w}_i = (w_{i,1}, ..., w_{i,m})$ using all his matching values, signs this matching response using ring signatures in Algorithm 2, and sends $\boldsymbol{w}_i = (w_{i,1}, ..., w_{i,m})$ and its ring signature $\boldsymbol{\sigma}_i = (\sigma_{i,1}, ..., \sigma_{i,d})$ to stranger $S$ (as shown in Fig. 3).

**Algorithm 2** $\mathsf{RingSign}$

Given all the group members' public keys $(pk_1, ..., pk_d) = (v_1, ..., v_d)$, a matching response $\boldsymbol{w}$, and a private key $sk_s = u_s$ for some $s$, this group member $u_s$

1) Randomly chooses $y_i \in Z_p$ and computes $\sigma_i = g_1^{y_i}$ for all $i \neq s$ and $i \in [1, d]$.
2) Computes $h = H(\boldsymbol{w}) \in G_1$ and sets
$$\sigma_s = \left( \frac{h}{\psi(\prod_{i \neq s} v_i^{y_i})} \right)^{1/u_s}, \quad (4)$$
where $H : \{0,1\}^* \to Z_p$ is a full-domain hash function and $\psi : G_2 \to G_1$ is a computable isomorphism.
3) Outputs the ring signature $\boldsymbol{\sigma} = (\sigma_1, ..., \sigma_d) \in G_1^d$.

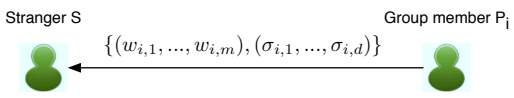



Fig. 3. Group member $P_i$ sends matching response $\boldsymbol{w}_i$ and its signature $\boldsymbol{\sigma}_i$ to stranger $S$.

***Match.*** Upon receiving a matching response $\boldsymbol{w}_i$ and its ring signature $\boldsymbol{\sigma}_i$, stranger $S$ first verifies the correctness of this matching response according to Algorithm 3. If the

matching response passes the verification, stranger $S$ decrypts each $w_{i,j} \in \boldsymbol{w}_i$ with decryption algorithm $\mathtt{Dec}$. If the result of decryption matches one of his attributes, then $a_{i,j}$ is a matched attribute. Otherwise, it is an unmatched attribute. This is because

$$\begin{aligned} \mathtt{Dec}(w_{i,j}) &= \mathtt{Dec}(\mathtt{Enc}(\tau_{i,j} \cdot P(a_{i,j}) + a_{i,j})) \\ &= \tau_{i,j} \cdot P(a_{i,j}) + a_{i,j}, \end{aligned} \tag{5}$$

where $P(a_{i,j}) = 0$ and $\mathtt{Dec}(w_{i,j}) = a_{i,j}$, if $a_{i,j} \in \mathcal{A}_s$.

**Algorithm 3** `RingVerify`

Given all the group members' public keys $(pk_1, ..., pk_d) = (v_1, ..., v_d)$, a matching response $\boldsymbol{w}$, and its ring signature $\boldsymbol{\sigma} = (\sigma_1, ..., \sigma_d)$, the stranger

1) Computes $h = H(\boldsymbol{w}) \in G_1$.
2) Verifies

$$e(h, g_2) \stackrel{?}{=} \prod_{i=1}^{d} e(\sigma_i, v_i). \tag{6}$$

If the equation holds, then this matching response is correct and signed by a group member. Otherwise, it is not.

After decrypting all the matching values from all the group members, stranger $S$ is able to calculate the matching degree $D_j$, for $j \in [1, k]$ and obtain group matching information $\mathcal{D}(\mathcal{P}) = (D_1, ..., D_k)$ about this group $\mathcal{P}$.

### C. Batch Verification

Generally, the stranger in Gmatch has to verify $d$ matching responses from all the $d$ group members separately, which introduces prohibitive huge computation cost to himself. Utilizing properties of bilinear maps, the stranger can reduce the cost of verification by checking the integrity of all the matching responses in a *batch* manner, instead of verifying them one by one. The details of batch verification are shown in Algorithm 4.

**Algorithm 4** `BatchVerify`

Given all the group members' public keys $(pk_1, ..., pk_d) = (v_1, ..., v_d)$, all the $d$ matching responses $(\boldsymbol{w}_1, ..., \boldsymbol{w}_d)$, and their ring signatures $(\boldsymbol{\sigma}_1, ..., \boldsymbol{\sigma}_d)$, where $\boldsymbol{\sigma}_i = (\sigma_{i,1}, ..., \sigma_{i,d})$, the stranger

1) Computes $h_l = H(\boldsymbol{w}_l) \in G_1$, for all $l \in [1, d]$.
2) Generates $d$ random number $(\lambda_1, ..., \lambda_d) \in Z_p^d$.
3) Verifies

$$e(\prod_{l=1}^{d} h_l^{\lambda_l}, g_2) \stackrel{?}{=} \prod_{i=1}^{d} e(\prod_{l=1}^{d} \sigma_{l,i}^{\lambda_l}, v_i). \tag{7}$$

If the equation holds, then all the matching responses are valid. Otherwise, they are not all valid.

Note that batch verification will fail if only one invalid matching response exists. To further detect a small number of invalid ones among all the responses, so the valid ones can still pass verification, we can leverage *binary search* [9] during batch verification. More specifically, when batch verification fails, the stranger further divides the set of all the matching responses into two halves, and rechecks each half using batch verification. If one half passes, all the matching responses in this half are valid. Otherwise, two sub halves of this half will be further rechecked until all the invalid ones are found.

### D. Higher Privacy Levels

There are two ways to modify the construction of Gmatch, so that it can achieve even higher privacy levels. First, similar to the previous work [2], each matching value is computed as $w_{i,j} = \mathtt{Enc}(\tau_{i,j} P(a_{i,j}))$ instead of $w_{i,j} = \mathtt{Enc}(\tau_{i,j} P(a_{i,j})) + a_{i,j}$. Then, when the decryption result is 0, it means that there is a matched attribute in the group. However, the stranger cannot determine which particular attribute in his profile is matched to this attribute.

Second, instead of signing the matching response $\boldsymbol{w}_i$, each group member signs each matching value $w_{i,j} \in \boldsymbol{w}_i$ one by one using ring signatures, and sends each matching value separately to the stranger. Then, the stranger believes that every matching value is correct and signed by a group member, but cannot distinguish whether two different matching values are from the same group member. Further, the stranger cannot tell whether two different matched attributes are from the same group member. However, to achieve this higher privacy level, each group member has to operate $m$ ring-signing operations instead of only one ring-signing operation, and the stranger also needs to verify $m \times d$ ring signatures in total, which will increase the computation cost of the entire scheme.

## V. Security Analysis

In this section, we show that Gmatch is able to achieve the privacy properties we defined in Section II.

***Theorem 1:*** Assuming that the additive homomorphic encryption is semantically secure, Gmatch achieves **stranger's attributes privacy**.

*Proof:* In Gmatch, group member $P_i$ obtains $k + 1$ encrypted coefficients of polynomial $P(x)$ computed by additive homomorphic encryption algorithm $\mathtt{Enc}$. If the additive homomorphic encryption $\mathtt{Enc}$ is semantically secure [8], it is computational infeasible for the group member to derive any plaintext when given only its corresponding ciphertext and public encryption key $pk_s$. Because Paillier cryptosystem, which we use in Gmatch, is semantically secure. Then, given encrypted coefficients $\{\mathtt{Enc}(\alpha_0), \ldots, \mathtt{Enc}(\alpha_k)\}$ and public encryption key $pk_s$, group member $P_i$ cannot learn $\{\alpha_0, \ldots, \alpha_k\}$ without the stranger's private key $sk_s$. Further, group member $P_i$ is not able to reconstruct the polynomial $P(x)$ and compute all the $k$ roots of $P(x)$. Therefore, all the $k$ attributes in stranger' profile are not revealed to any group member, stranger's attributes privacy is achieved. ■

***Theorem 2:*** Assuming parameter $\tau_{i,j}$ for matching value $w_{i,j}$ is *random*, Gmatch achieves **group members' attributes privacy**.

*Proof:* According to Equation (5), the decryption result of matching value $w_{i,j}$ can be described as follows:

$$a_{i,j} + \tau_{i,j} \cdot P(a_{i,j}) = \begin{cases} a_{i,j}, & if \quad P(a_{i,j}) = 0 \\ random, & if \quad P(a_{i,j}) \neq 0 \end{cases} \quad (8)$$

Clearly, when $a_{i,j}$ is a matched attribute, it is a root of polynomial $P(x)$, then we have $P(a_{i,j}) = 0$, and the decryption result is $a_{i,j}$. When $a_{i,j}$ is an unmatched attribute, if $\tau_{i,j}$ is a random number, we have $P(a_{i,j}) \neq 0$, and the decryption result is a random value. Therefore, what the stranger obtains after decryption is either an attribute in his profile or a random value that does not disclose any unmatched attribute in any group member's profile. ■

***Theorem** 3:* Assuming each matching response is signed by ring signatures, then Gmatch achieves **exact matching information privacy** with probability $1 - \frac{1}{d!}$, where $d$ is the size of the group.

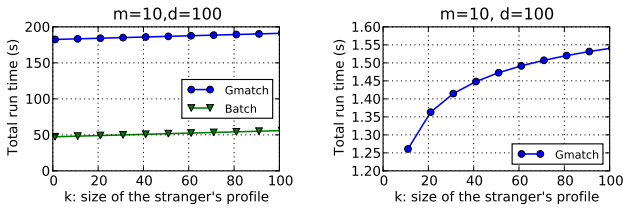
*Proof:* Due to the properties of ring signatures in BGLS [4], when verifying a matching response, the stranger is convinced that this matching response is signed by a group member but cannot distinguish which particular member it is from. The stranger can successfully distinguish that a matching response belongs to a particular group member with a probability of $1/d$. Since the total number of matching responses received by the stranger is $d$, the total probability that the stranger successfully discloses the exact matching information between himself and every group member is $\frac{1}{d!}$. Therefore, Gmatch can achieve exact matching information privacy with probability $1 - \frac{1}{d!}$. ■

As we analyzed in Theorem 2, during the group matching, unmatched attributes in group members' profiles are not disclosed to the stranger. However, by honestly following the group matching, the stranger can still obtain more information than allowed by performing all zero polynomial attacks [1]. More specifically, the stranger sets all $k + 1$ coefficients of polynomial $P(x)$ as zeros. Under this type of attacks, the computation result of $P(a_{i,j})$ is always zero, which makes the random number $\tau_{i,j}$ useless. Then, all the decryption results of matching values are attributes from one of group members' profiles. In this case, the stranger is able to learn all the attributes in all group members' profiles. Making matters worse, because the stranger only sends the encrypted coefficients to each group member, and the encryption algorithm is *probabilistic*, group members cannot check whether those coefficients are all zeros or not. To prevent this type of attacks, we set one of the $k + 1$ coefficients as 1, and is sent to group members without encryption. Similar methods can also be found in [1] [10].

## VI. Performance

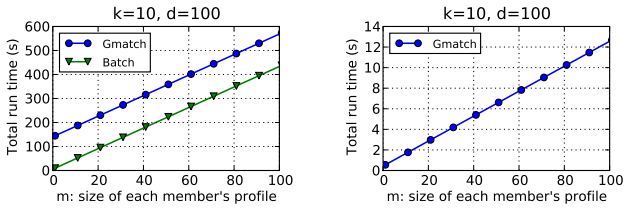
We now evaluate the efficiency of Gmatch in experiments by using the PBC library. All the experiments are tested on a 2.26 GHz Linux system. For the ease of implementation, we assume $G_1 = G_2$. The elliptic curve we used is an MNT curve with a base field size of 159 bits. The length of each element of $G_1$ is $|p| = 160$ bits, and the length of an element of $G_T$ is 1024 bits. An encrypted coefficient under Enc is an element of $Z_n$, where $|n| = 2048$ bits.

*1) Efficiency of Gmatch:* As we can see from Fig. 4(a), Fig. 5(a) and Fig. 6(a), the efficiency of group matching can be significantly improved by utilizing batch verification. More specifically, when the size of users' profiles are fixed in Fig. 6(a), the rum time of Gmatch without batch verification exponentially increases with the total number of group members, while the one with batch verification only increases linearly with the group size.



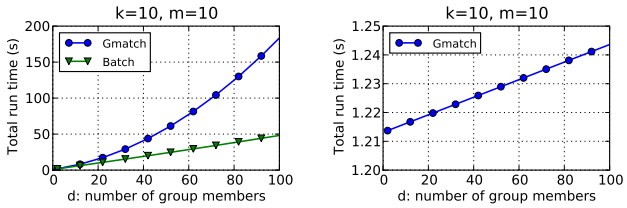

(a) Run time at stranger $S$. (b) Run time at group member $P_i$.

Fig. 4. Impact of $k$ on the run time, where $m = 10$ and $d = 100$




(a) Run time at stranger $S$. (b) Run time at group member $P_i$.

Fig. 5. Impact of $m$ on the run time, where $k = 10$ and $d = 100$




(a) Run time at stranger $S$. (b) Run time at group member $P_i$.

Fig. 6. Impact of $d$ on the run time, where $m = 10$ and $k = 10$

The efficiency of group matching at each group member are illustrated in Fig. 4(b), Fig. 5(b) and Fig. 6(b). The run time at each group member in Gmatch is greatly increasing with the size of each group member's profile, but hardly affected by the size of the stranger's profile or the size of the group.

*2) Efficiency of Batch Verification with Invalid Matching Responses:* We now evaluate the performance of batch verification under different numbers of invalid matching responses. Clearly, the increasing number of invalid responses will reduce the efficiency of batch verification. In this experiment, we set the total number of matching responses $d = 100$ and assume it

always requires the *worst-case* algorithm to detect invalid ones from all the matching responses. As shown in Fig. 7, when less than $10\%$ of all the matching responses are invalid, batch verification is still efficient than verifying them separately.

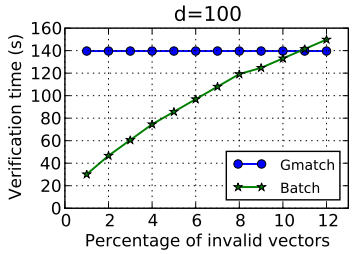



Fig. 7. Comparison on verification time between batch verification and one-by-one verification where $d = 100$.

## VII. Related Work

### A. Two-party private matching

Freedman *et al.* [2] proposed a private matching scheme, which allows a client and a server compute the set intersection with their own private sets. During private matching, the client only obtains the set intersection while the server does not know any matching result. Agrawal *et al.* [6] introduced a private matching scheme between two databases using commutative encryptions. Hazay and Lindell [7] exploited pseudo random functions to evaluate set intersection. In [11], Dachman-Soled *et al.* exploited polynomial evaluations to compute the set intersection between two parties, and also leveraged shamir secret sharing and cut-and-choose protocol to improve efficiency. Recent work in [12] introduced an authorized private set intersection (APSI) based on blind RSA signatures. In APSI, each element in the client's set must be authorized by some mutually trusted authority.

### B. Multi-party private matching

Kissner and Song [13] proposed a multi-party private matching scheme to compute the union, intersection and element reduction operations for multiple sets. However, this scheme requires a group decryption among multiple entities, which is impractical between the stranger and group members in social networks. Ye *et al.* [14] extended previous scheme to a distributed scenario with multiple servers. The dataset of the original server is shared by several sub-servers using $(t, w)$-shamir secret sharing. Therefore, any $t-1$ or fewer sub-servers cannot discover the dataset of the original server. Sang *et al.* [15] improved the efficiency of private matching among multiple parties by exploiting an extra $N \times N$ nonsingular matrix, where $N$ is the total number of entities. Li and Wu [10] proposed a private multi-party set intersection scheme based on the two-dimensional verifiable secret sharing scheme.

### C. Private matching in social networks

FindU [1] focuses on finding the *best matched* user from the group in mobile social networks. Yang *et al.* [16] introduced E-SmallTalker, which allows users to privately match other people in mobile social networks using the iterative bloom filter (IBF) protocol.

## VIII. Conclusion

In this paper, we proposed Gmatch, a secure and privacy-preserving group matching in social networks. With Gmatch, the stranger can successfully collect group matching information while the private information of group members are preserved. Our experimental results show that Gmatch can efficiently compute correct group matching information with batch verification.

## Acknowledgement

We are grateful to the anonymous reviewers for their helpful comments. This work is supported by the National Science and Technology Major Project (No. 2012ZX03002003), Fundamental Research Funds for the Central Universities (No. K50511010001), National 111 Program (No. B08038), Doctoral Foundation of Ministry of Education of China (No. 20100203110002) and Program for Changjiang Scholars and Innovative Research Team in University.

## References

[1] M. Li, N. Cao, S. Yu, and W. Lou, "FindU: Private-Preserving Personal Profile Matching in Mobile Social Networks," in *Proc. IEEE INFOCOM*, 2011, pp. 2435 – 2443.
[2] M. J. Freedman, K. Nissim, and B. Pinkas, "Efficient Private Matching and Set Intersection," in *Proc. EUROCRYPT*. Spring-Verlag, 2004, pp. 1–19.
[3] R. L. Rivest, A. Shamir, and Y. Tauman, "How to Leak a Secret," in *Proc. ASIACRYPT*. Springer-Verlag, 2001, pp. 552–565.
[4] D. Boneh, C. Gentry, B. Lynn, and H. Shacham, "Aggregate and Verifiably Encrypted Signatures from Bilinear Maps," in *Proc. EUROCRYPT*. Springer-Verlag, 2003, pp. 416–432.
[5] A. Sorniotti and R. Molva, "Secret Interest Groups (SIGs) in Social Networks with an Implementation on Facebook," in *Proc. ACM SAC*, 2010, pp. 621–628.
[6] R. Agrawal, A. Evfimievski, and R. Srikant, "Information Sharing Across Private Databases," in *Proc. ACM SIGMOD*, 2003, pp. 86–97.
[7] C. Hazay and Y. Lindell, "Efficient Protocols for Set Intersection and Pattern Matching with Security Against Malicious and Covert Adversaries," in *Proc. TCC*. Springer-Verlag, 2008, pp. 155–175.
[8] P. Paillier, "Public-Key Cryptosystems Based on Composite Degree Residuosity Classes," in *Proc. EUROCRYPT*. Springer-Verlag, 1999, pp. 223–238.
[9] C. Wang, Q. Wang, K. Ren, and W. Lou, "Privacy-Preserving Public Auditing for Data Storage Security in Cloud Computing," in *Proc. IEEE INFOCOM*, 2010, pp. 525–533.
[10] R. Li and C. Wu, "An Unconditionally Secure Protocol for Multi-Party Set Intersection," in *Proc. ACNS*. Springer-Verlag, 2007, pp. 226–236.
[11] D. Dachman-Soled, T. Malkin, M. Raykova, and M. Yung, "Efficient Robust Private Set Intersection," in *Proc. International Conference on Applied Cryptography and Network Security*. Springer-Verlag, 2009, pp. 125–142.
[12] E. D. Cristofaro and G. Tsudik, "Practical Private Set Intersection Protocols with Linear Complexity," in *Proc. FC*, 2010, pp. 143–159.
[13] L. Kissner and D. Song, "Privacy-Preserving Set Operations," in *Proc. CRYPT*. Springer-Verlag, 2005, pp. 241–257.
[14] Q. Ye, H. Wang, and J. Pieprzyk, "Distributed Private Matching and Set Operations," in *Proc. ISPEC*. Springer-Verlag, 2008, pp. 347–360.
[15] Y. Sang, H. Shen, Y. Tan, and N. Xiong, "Efficient Protocols for Privacy Preserving Matching Against Distributed Datasets," in *Proc. ICICS*. Springer-Verlag, 2006, pp. 210–227.
[16] Z. Yang, B. Zhang, J. Dai, A. C.Champion, D. Xuan, and D. Li, "E-SmallTalker: A Distributed Mobile System for Social Networking in Physical Proximity," in *Proc. IEEE ICDCS*, 2010, pp. 468 – 477.