# S-MATCH: Verifiable Privacy-preserving Profile Matching for Mobile Social Services

Xiaojing Liao, Selcuk Uluagac, Raheem A. Beyah
School of Electrical and Computer Engineering, Georgia Institute of Technology
xliao@gatech.edu, {selcuk, rbeyah}@ece.gatech.edu

***Abstract*—Mobile social services utilize profile matching to help users find friends with similar social attributes (e.g., interests, location, background). However, privacy concerns often hinder users from enabling this functionality. In this paper, we introduce S-MATCH, a novel framework for *privacy-preserving profile matching* based on *property-preserving encryption* (PPE). First, we illustrate that PPE should not be considered secure when directly used on social attribute data due to its key-sharing problem and information leakage problem. Then, we address the aforementioned problems of applying PPE to social network data and develop an efficient and verifiable privacy-preserving profile matching scheme. We implement both the client and server portions of S-MATCH and evaluate its performance under three real-world social network datasets. The results show that S-MATCH can achieve at least one order of magnitude better computational performance than the techniques that use homomorphic encryption.**

***Keywords*-profile matching; privacy; property-preserving encryption; symmetric encryption;**

## I. Introduction

With the explosive growth of social networks and mobile devices, mobile social service has become a popular method among traditional online social network users to build social relationships and to share interests. For instance, Groupon, Yelp, Wechat are among the most popular mobile social services in the market, through which users can share their location information, share their interests, and chat with friends nearby. To build social relationships and share interests, *profile matching* is a fundamental and significant step for mobile social services. During the profile matching process, users' social profiles are compared in an outsourced server owned by social network application providers to find other users with similar social attributes (e.g., interests, location, background) in the social networks. An effective profile matching process benefits users and social network providers alike. Users enjoy a more pleasant experience while social network providers see increased user activities in their social networks.

However, social profile attributes used in the profile matching process include sensitive information about users. For instance, even though the 'public' attributes such as an Interest (or 'Like') in Facebook are regarded as harmless, they can reveal some basic 'private' social attributes (e.g., age, health condition, or religion [1]). In mobile social services, the violation of the privacy of the users' social profiles can pose serious problems. Using social profile information, attackers can flood users with unwanted advertisements [2] or easily crack weak user passwords [3]. Hence, the privacy concerns must be addressed when developing profile matching techniques for mobile social networks [4], [5]. In addition to security, clients of mobile social networks run on resource-constrained mobile devices. Therefore, a *privacy-preserving* and *power-efficient* profile matching scheme is needed for mobile social services.

To preserve the privacy of the profile matching computation, homomorphic encryption [6], [7] has been widely used for privacy-preserving profile matching [8]–[12]. In homomorphic encryption, certain types of computations are allowed on ciphertext and the computation results retain some relationship among plaintexts. For instance, in homoPM [8], after users encrypt plaintexts, which are blinded by a random number $\delta$, with homomorphic encryption and upload the ciphertexts onto the server of the mobile social networks, the server can conduct some computations (e.g., modular multiplication) on the ciphertexts and obtain the comparison relationship among plaintexts. However, while functional, homomorphic encryption is *not* practical for mobile social services because of the two following reasons: First, homomorphic encryption is computationally intensive and slow. Second, the schemes based only on homomorphic encryption faces the *result verification problem* that fake profile matching results can be sent to the querying user from the server without being detected by the user.

A better candidate for the privacy-preserving profile matching process for mobile social services is *property-preserving encryption* (PPE). A naive approach utilizing PPE [13] to match the profile secretly is that each user encrypts their social attributes with the PPE separately and sends all of the encrypted attributes to the server. As the ciphtexts preserve some information of the plaintexts such as relative distance and order between the plaintexts in PPE, comparison operations on the ciphertexts are possible. Therefore, the untrusted server is able to process the profile matching algorithms based on comparison operations without knowing the plaintexts. However, as we analyze in Section IV, PPE can cause high information leakage if it is directly used on low entropy social attribute data, and the secret profile attribute values can be deciphered. Also, to guarantee the property such as relative distance and order among all the ciphertexts, the plaintexts should be encrypted by the same key. However, it is not practical for all the users share the same secret key. Because when a malicious user colludes with the untrusted server, all the data face the threat of plaintext recovery.

In this paper, we propose S-MATCH, a privacy-preserving profile matching technique for mobile social services in which the privacy-preserving operations are achieved utilizing PPE. The main contributions of the paper are summarized as follows:

- We show the challenges of using PPE directly on social attribute data for profile matching. Specifically, we highlight the key-sharing problem and the information leakage problem by analyzing the entropy of three real-world datasets.
- We address the information leakage problem by providing a technique to increase the entropy of the user profiles so that PPE can be used in our privacy-preserving profile matching scheme. Also, we present a key generation protocol for users, which addresses the key-sharing problem.
- We present a verification protocol for users to verify the profile matching results, but learn nothing about other users' profile attributes.
- We show the provable security of S-MATCH, which indicates that S-MATCH is protected from plaintext recovery under ordered known plaintext attack and known key attack (i.e., PR-OKPA and PR-KK).
- We demonstrate a prototype implementation of S-MATCH with an application on an Android-based mobile testbed and evaluate its performance using three real-world datasets. Our results indicate the efficiency and validity of our scheme.

This paper proceeds as follows. We discuss the related work in Section II. In Section III, we describe the cryptographic primitives in this work. We present the challenges of using PPE for privacy-preserving profile matching in Section IV. In Section V, we show the problem formulation. In Section VI, our approach is described in detail. Section VII presents the security analysis of S-MATCH. In Section IX, our scheme is evaluated using real-world datasets and compared with a representative scheme based on homomorphic encryption. Finally, in Section X, the conclusion and future work are presented.

## II. Related work

**Private Profile Matching.** Profile matching is critical for social networks. Recently, Zhang et al. [14] improved the performance of the earlier profile matching schemes by introducing the symmetric cryptosystem to conduct operations. Also, the scheme they proposed is verifiable. However, the scheme is designed in the two-party matching scenario, which introduce large communication cost when extended to a profile matching scheme in large scale. In [9], a privacy-preserving personal profile matching scheme was proposed, where minimal information about the users' social attribute is exchanged. In [10], Arb et al. proposed a mobile social networking platform called VENETA to secretly detect friends of friends. In [11], Wang et al. proposed a secure and privacy-preserving social networks group matching scheme Gmatch. In [15], Nagy et al. developed an efficient friend-finder application. However, all of these schemes are attribute-level profile matching based on *Private Set Intersection* [16], [17], which means that they are not able to differentiate users with different attribute values. Zhang et al. [8] proposed a fine-grained profile matching protocol to differentiate users with different attribute values. Li et al. [12] improved the scheme in [8] by introducing a novel blind vector transformation technique to protect the profile matching process against the runaway attack. These profile matching schemes are conducted through homomorphic encryption [6], [7] such as Paillier's cryptosystem [18]. However, the privacy-preserving profile matching schemes only based on homomorphic encryption are too computationally expensive for mobile devices and are not verifiable. Table I shows the difference of our paper from the previous works. The category of the schemes includes symmetric encryption (SE) and homomorphic encryption (HE). Security of the schemes includes malicious model (M) and honest-but-curious model (HBC). Verification means that the profile matching results from the server are verifiable. Fine-grained match means that the profile matching is processed on the attribute-value-level, compared to the attribute-level. And fuzzy match means that the profile matching results include not only the perfect matching result but also the top-k matching results. As illustrated by Table I, S-MATCH is the most full-featured and efficient scheme.

**Security of OPE.** Motivated by the first OPE scheme proposed by Agrawal et al. [19], Boldyreva [20] initiated the cryptographic study of OPE and first present the security goal of OPE, i.e., releasing nothing but order (indistinguishability under ordered chosen plaintext attack (IND-OCPA)). In [20], Boldyreva et al. indicated that IND-OCPA cannot be achieved. They proved that the OPE scheme they proposed was POPF-CCA secure (i.e., pseudorandom order-preserving function under chosen-ciphertext attack), which means that the adversary cannot distinguish the OPE scheme from the random order

TABLE I
A comparison of Related Works with our S-Match.

| | S-MATCH | ZLL13 [14] | ZZS12 [8] | LCY11 [9] | NCD13 [15] | LGD12 [12] |
|---|---|---|---|---|---|---|
| Category | SE | SE | HE | HE | HE | HE |
| Security | M/HBC | M/HBC | HBC | HBC | HBC | HBC |
| Verification | ✓ | ✓ | ✗ | ✗ | ✗ | ✗ |
| Fine-grained Match | ✓ | ✗ | ✓ | ✗ | ✗ | ✓ |
| Fuzzy Match | ✓ | ✓ | ✓ | ✗ | ✗ | ✓ |

preserving function. mOPE proposed by Popa et al. [21] was the first OPE scheme to achieve IND-OCPA. However, mOPE is an interactive scheme, which is not suitable for the privacy-preserving profile matching scenario in our work.

## III. Cryptographic Primitives

***PPE.*** A deterministic property-preserving encryption (PPE) is a tuple $PPE = (Keygen, Enc, Dec)$. In PPE, the ciphertexts preserve the property $P$ of a set of plaintexts $M$ such as distance or order. To encrypt a set of plaintexts $\{m_i : m_i \in M\}$, the user generates a PPE key $K \overset{\$}{\leftarrow} Keygen$ and the ciphertexts $\{c_i \leftarrow Enc(K, m_i) : m_i \in M\}$. The property $P$ which the ciphertexts preserve is formally defined as follows:

*Definition 1:* $PPE$ is an encryption scheme, which has property $P$ with parameters of fixed number $k$ and a publicly computable algorithm `Test`, such that

$$\texttt{Test}(c_1, \ldots, c_k) = P(m_1, \ldots, m_k)$$

where $c_i$ is the ciphertext of the plaintext $m_i$, $i \in \{1, \ldots, k\}$.

Order-preserving symmetric encryption (OPE) [19]–[21] is an example of a PPE with the property of order and $k = 2$. OPE, which was first proposed by Agrawal et al. [19], preserves the order of plaintexts and allows any comparison operations to be applied on ciphertexts. Given plaintexts $m_i, m_j$, corresponding ciphertexts $c_i, c_j$ satisfy the relationship that $m_i \geq m_j \Rightarrow c_i \geq c_j$. Distance-preserving encryption (DPE) proposed by Ozsoyoglu et al. [22] is an another example of a PPE scheme. DPE preserves the distance of the two numeric data after encryption such that for any three values $m_i, m_j, m_k$, $|m_i - m_j| \geq |m_j - m_k| \Rightarrow |c_i - c_j| \geq |c_j - c_k|$ and $k = 3$. PPE techniques are widely used in database indexes over encrypted tables [19], identifying similarities on sensitive data [23], and spam-email detection [24].

***OPRF.*** An oblivious pseudo-random function scheme is a tuple $OPRF = (Keygen, F)$. In OPRF, the user is able to obtain a pseudo-random number $r$, but the random number generator learns nothing about the user input $m$ and the pseudo-random number $r$. To generate the pseudo-random number $r$, a public key and a secret key are generated $(pk, sk) \overset{\$}{\leftarrow} Keygen$, where the random number generator owns the secret key $sk$ and the users utilize public key $pk$. An OPRF is an interactive protocol, and a pseudo-random number $r \leftarrow F(sk, m)$ is generated on the user side after a round of secure communication with the random number generator.

RSA-OPRF is an example of an OPRF. The key generation function utilizes RSA key generation results $((N, e), (N, d))$, where $ed \equiv 1 \ mod \ \phi(N)$ and outputs $(N, d)$ as the public-secret key pair. A pseudo-random number $r$ is generated by first hashing the input $m$ and encoding it as $x = h(m) \cdot s^e \ mod \ N$, where $s$ is a random number and $h()$ is a hash function. After interacting with the random number generator and obtaining $y = x^d \ mod \ N$, the user outputs $r = h'(y \cdot s^{-1} \ mod \ N)$ as the pseudo-random number, where $h'()$ is another hash function.

## IV. Challenges of Using PPE for Private Profile Matching

In this section, we first present the key sharing problem and the information leakage problem of PPE, which is exacerbated when the message space is small. Then, we indicate that the social networks data have low entropy and landmark attributes by analyzing three real-world social networks datasets. Finally, we conclude PPE cannot be directly used to encrypt social networks data.

### A. Key Sharing in PPE

Similar to the homomorphic encryption, the ciphertexts can only preserve the property of the plaintexts if the plaintexts are encrypted by the same property-preserving key in PPE. Even though there is not an all-accessible public key like that in homomorphic encryption, sharing of the same property-preserving secret key among all users in PPE is still unacceptable and not practical. In the worst case, when an honest-but-curious user colludes with the untrusted server, all the users' data will be leaked. Hence, the key sharing problem should be addressed, when PPE is utilized for privacy-preserving profile matching.

### B. Information Leakage of PPE

In PPE, ciphertexts leak the property information associated with the plaintexts, which makes it vulnerable when the number of the plaintexts is limited. For example, in OPE, an attacker can learn the order of the plaintext from the ciphertext, making a chosen-ciphertext attack significantly easier. Furthermore, when the number of the plaintexts encrypted by OPE is small, the vulnerability is exacerbated. Assume that there is an untrusted server with $n$ pairs of known plaintexts and ciphertexts along with the set of ciphertexts it stored. Given a plaintext $p_i$, the untrusted server can recover $c_i$ from the decryption $D(c_i, k)$ using the known plaintext-ciphertext pairs and by analyzing the property, where $D(c_i, k)$ is the decryption of the PPE with ciphertext $c_i$ and key $k$.

We illustrate this with a simple example shown in Figure 1 assuming that an OPE scheme is utilized. In our illustration, an untrusted server tries to obtain the ciphertext of 5 using the known ciphertext-plaintext $(c_i, p_i)$ pairs of $(30, 3)$ and $(70, 7)$. As the ciphertexts stored in the untrusted server are derived from the OPE scheme, the order relationships among the ciphertexts can be easily obtained by the server. Then, the untrusted server is able to prune the search space to find the ciphertext of 5 by analyzing the property of the ciphertext values (e.g., order). With ciphertext-plaintext pairs $((30, 3)$ and $(70, 7)$, the server can infer that the ciphertext of 5 is between 30 and 70. As a result, the size of the search space is 3 (Figure 1(a)). However, the search space is larger, 39, for a configuration with more entries as shown in Figure 1(b). Hence, when the number of plaintexts is small, the time to break the ciphertext and obtain the corresponding plaintexts is shorter than that when the number of plaintext values is large.

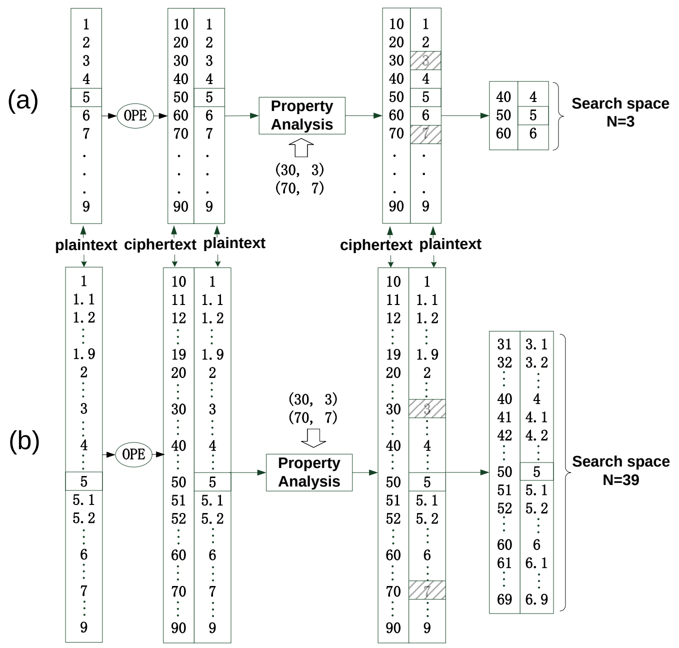


Fig. 1. A simple illustration of information leakage of OPE schemes, where (30,3) and (70,7) are the known ciphertext-plaintext pairs. The untrusted server tries to obtain the ciphertext of 5 (plaintext) analyzing the order property.

## C. Low Entropy of Social Networks

In this sub-section, we show the low entropy associated with social network datasets. For this, we analyze three real-world datasets and present how the limited number of attribute values and the existence of *landmark* attribute [1], [25] facilitate the leakage of information.

The three real-world datasets that are analyzed are as follows: (1) *Infocom06* dataset [26] was from the attendees of the IEEE Infocom 2006 Conference, who contributed their mobility information ((x,y)-coordinate data) through their mobile devices and social attributes from questionnaires. (2) *Sigcomm09* dataset [27] was collected by smartphones distributed to a set of volunteers in the ACM Sigcomm 2009 Conference. Each device was initialized with some basic (e.g., country, affiliation) and extended social profile (e.g., interests from Facebook profile). Also, the devices recorded the locations of the volunteers during the conferences. (3) *Weibo* dataset [28] was captured through the user profile API and keyword extra API in Sina Weibo (Chinese twitter), which include some basic and extended social attributes (i.e., 10 interests). The user interest attribute is defined as the frequency of semantically related keywords. Also, the 'check-in' information includes location information based on Google map locations, which is an interface provided by Weibo.

First, as discussed in Section IV-B, PPE will leak more information due to the limited number of plaintexts. Now, we utilize the entropy to evaluate the limited number of the attribute values in social profile data so that we are able to determine whether PPE can be directly used to encrypt the social profile attributes. In order to show the entropy of a given social attribute $\mathcal{A}_l$ in a social network, we calculate

$$H(\mathcal{A}_l) = -\sum_i \frac{T_i}{U} \log \frac{T_i}{U} \tag{1}$$

where $T_i$ is the number of users with the attribute value $i$, $U = \sum_i T_i$ is the total number of users, and $\frac{T_i}{U}$ indicates the probability of attribute $\mathcal{A}_l$ having value $i$. $H(\mathcal{A}_l)$, therefore, represents the entropy of social attribute $\mathcal{A}_l$.

The properties of Infocom06, Sigcomm09, and Weibo datasets are summarized in Table II. We note that attributes with significantly small entropy exist in the real-world social profile datasets. This situation aggravates the information leakage of a PPE scheme.

Second, we show how the landmark attributes [1], [25] will lead to information leakage. A landmark attribute is based on the fact that human activities (e.g., geographic and social constrains [29]) exhibit structured patterns. For example, some geographic locations may have much more 'check-ins' than others. In other words, landmark attributes are prevalent in social attribute data. Since PPE is a symmetric encryption and some users should share a key to encrypt the profile, the ciphertexts of the landmark attributes are still noticeable. We formally define the *landmark attribute* as follows:

*Definition 2: Landmark attribute* is an attribute with value $i$ whose probability $\frac{T_i}{U}$ larger than threshold $\tau$, where $T_i$ is the number of users with the attribute value $i$ and $U$ is the total number of users, $U = \sum_i T_i$.

As also discussed in [1] and [25], the existence of landmark attributes in social profile data can undermine the anonymization, which increases the social profile dataset's propensity for leakage. For example, when an untrusted server observes a ciphertext appearing more often than others, the untrusted server can regard this as ciphertext generated by the encryption of a landmark attribute value. The existence of landmark attributes associated with the three real-world datasets is shown in Table II. As seen, for each dataset, at least one or more landmark attribute exist, which can exacerbate the usage of PPE schemes in social network settings. As Table II

TABLE II
THE PROPERTIES OF DATASETS

| Dataset | Node | the Number of Attributes | Entropy | | | Landmark Attribute | |
|---|---|---|---|---|---|---|---|
| | | | AVG | MAX | MIN | $\tau = 0.6$ | $\tau = 0.8$ |
| Infocom06 | 78 | 6 | 3.10 | 5.34 | 0.82 | 2 | 1 |
| Sigcomm09 | 76 | 6 | 3.40 | 5.62 | 0.86 | 3 | 1 |
| Weibo | 1 million | 17 | 5.14 | 9.21 | 0.54 | 5 | 3 |

indicates, real-world social profile datasets have low entropy and landmark values, which leads to high information leakage and the situation is aggravated when these datasets are used with PPE.

In conclusion, as Table II indicates, real-world social profile datasets have low entropy and landmark values, which leads to high information leakage with PPE. Therefore, the user profile attribute data used in a privacy-preserving profile matching scheme should have high entropy. Accordingly, in Section VI, we propose a technique to increase the entropy to enable private profile matching based on PPE.

## V. Problem Formulation

In this section, the system model and assumptions are given. Then, we outline the adversary model and design goals.

### A. System Model and Assumptions

A privacy-preserving profile matching scheme involves users with mobile devices running the same mobile social services and an untrusted server to process profile matching operations. Each user has a social profile, including some social profile attributes such as gender, education, location, and interests. Social attribute data can be generated through three methods: user input in online social networks (e.g., birthday, gender), device capture using sensors (e.g., location), and data analysis based on the user behavior in online social networks (e.g., interests). For instance, several methods have been proposed to extract user interest information from Facebook's 'Like' using semantic knowledge based techniques [1], [30], [31].

We assume each user $v$ updates her encrypted social profile $\{c_i^{(v)}\}$ on the untrusted server periodically. At another time $t$, user $v$ submits a query request for profile matching, $Q_q = < q, t, ID_v >$, to an untrusted server, where $q$ is the query ID, $t$ is the time-stamp and $ID_v$ is the identity of user $v$ in a mobile social service. Then, the untrusted server correctly matches profiles using the encrypted user profiles, and returns $k$ nearest profiles matching results to the user. Finally, the user verifies whether the profile matching results are correct.

Without loss of generality, we assume that each user has a unique ID and share the same social profile format, where each attribute value $a_i \in Z_n$ . Also, each mobile device has similar storage capacity and computing power (e.g., smartphones). Each untrusted server is powerful and resourceful enough to store social profile attribute data and process profile matching operations (e.g., requests). As previous works [8], [9], [12], [14], [15], we assume that social profile attribute data in the mobile device are captured from trusted social network interfaces such that users would not change their social profile attribute data.

### B. Adversary Model and Design Goals

In our threat model, we consider the following three types of adversaries regarding the server and users:

- *Honest-but-curious server* where the server follows the designated protocol specification honestly while it is curious to analyze data in its storage so as to learn additional information of the plaintext beyond the property defined in the PPE. For example, it can execute a chosen-ciphertext attack where it chooses a ciphertext and obtain its corresponding plaintext. Then, it can enumerate all the possible attribute values to determine the plaintext of the attribute values for each users. Such a server can use this type of data for targeted ad campaigns.
- *Honest-but-curious user* where the user acts in an 'honest' fashion to obtain the correct profile matching results, but 'curious' fashion to obtain other users' exact profile attribute data by eavesdropping the communications or colluding with the untrusted server. Such a user can use this type of data to impersonate other users at a later time.
- *Malicious server* where a compromised server does not follow the designated protocol but returns fake profile matching results to the user. Similar to the works in the literature [8], [9], [12], [14], [15], the service provider would not allow the user outside to access the dataset or affect the computation results for security reasons and public reputation reasons.

To address the adversary models above, a privacy-preserving profile matching scheme for mobile social services is proposed. Our scheme, S-MATCH, achieves security and performance guarantees as follows:

- *Privacy*: The untrusted server does not learn additional information of the users' social profiles. Specifically, even if the server knows that an intercepted ciphertext has plaintext from a set of known plaintext range, a secure scheme will prevent the attacker from extracting the plaintext. Thus, the scheme is protected from plaintext recovery attack. Also, each user cannot obtain other users' profiles.
- *Verification*: The profile matching result that indicates a match is for users with similar social profiles. Fake profile matching results from the server are detected.
- *Performance*: The above goals for privacy and verification should be power efficient with low computation overhead on resource-constrained mobile devices.

## VI. S-MATCH: Matching Profiles secretly

In this section, we introduce the details of the proposed privacy-preserving profile matching scheme after showing the overview of the system.

### A. System Overview

Our proposed scheme has three steps. First, users bootstrap the privacy profile matching process by increasing the entropy so that the privacy-preserving profile matching is processed on the high-entropy attribute chains encrypted by the PPE scheme. Then, each user commits her profile information and user ID to the server with authentication information $ciph_i$. Second, after obtaining the profile matching request

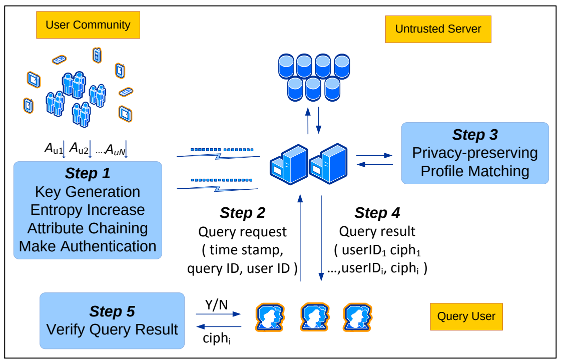


Fig. 2. Overview of the PPE-based privacy-preserving profile matching scheme.

$Q_q =< q, t, ID_v >$, the untrusted server conducts the efficient privacy-preserving profile matching process, then returns the matching result $R_q =< q, t, ID_1, ciph_1, \ldots, ID_k, ciph_k >$ to user $v$, where $q$ is the query $ID$, $t$ is the time-stamp, $ID_i$ is the identity of matching user $i$ in the mobile social service, and $ciph_i$ is user $i$'s authentication information for the querying user to verify the query results. Finally, with the authentication information $ciph_i$ of user $i$, the querying user $v$ can verify whether the profile matching result, which states user $i$ is a match, is correct or not. The overview of S-MATCH is shown as in Figure 2.

**Key Generation.** To address the key sharing problem while guaranteeing correct profile matching using ciphertext, we propose a fuzzy key generation scheme to generate the keys for OPE. In our scheme, the users with similar profiles will generate the same key. The fuzzy key generation solves the key sharing problem and improves the efficiency of the privacy-preserving scheme.

For this, we utilize a cyclic error-correcting decoding function (e.g., the Reed-Solomon decoding function [32]–[34]) to generate a profile key, where the users with the Euclidean-distance close profiles will generate the same profile key. Specifically, the profile of user $v$ is decoded by a Reed-Solomon decoder (RSD) to obtain a fuzzy vector $T^{(v)}$, and the profile key is generated by the fuzzy vector $T^{(v)}$, i.e., $K_{vp} = H(T^{(v)})$, where $H()$ is a one-way hash function. With RSD, the Euclidean-distance close profiles (i.e., $\parallel A_u - A_v \parallel \leq \theta$, where $\theta$ is the threshold of the RS Decoder) will be transformed to the same fuzzy vector and the users with the close profiles will generate the same profile key. To protect the key generation scheme from an offline brute-force attack, the profile key $K_{vp}$ is encoded as a pseudo-random number by OPRF.

*Definition 3:* Given the ordered series of user $u$ and user $v$'s attribute values, denoted by $A_u$ and $A_v$, the Euclidean distance of the profiles between user $u$ and user $v$ is defined as follows:

$$\parallel A_u - A_v \parallel = MAX\{a_i^{(u)} - a_i^{(v)}\}, \tag{2}$$

where $i \in \{1, 2 \ldots d\}$ and $a_i^{(u)}$ is the attribute value of user $u$.

The Reed-Solomon cyclic error-correcting codes (*RS code*) are constructed using Galois Fields and source symbols are viewed as coefficients of a polynomial over the Galois Fields. The RS code we are interested in are (n,d)-codes, where $d$ is the number of attribute values as the source symbols, and $n = 2^{10}$ as Galois Field $GF(10)$ is utilized.

**Entropy Increase.** To reduce the information leakage, the mobile user increases the entropy of each attribute while guaranteeing that the probability distributions of the profile attributes are uniform. The idea behind this technique is to construct a big-jump mapping of the attribute values with equal probability, where one profile attribute value is mapped to $N$ values to increase the uncertainty. For the big-jump mapping function $f$, there exists some big jumps $f(i+1) - f(i) \geq f(i) - f(j)$, where $1 \leq j < i - 1$. To quantify the privacy of the scheme, we use a $k$-bit-binary string to represent the value of the attribute. In other words, the message space is $2^k$. For the profile attribute $\mathcal{A}_i$ with $n_i$ possible values $a_j^{(i)}$ and the corresponding probability $p_j^{(i)}$, where $j \in \{1, 2 \ldots n_i\}$, each attribute value $a_j^{(i)}$ is mapped to $p_j^{(i)} \Delta$ binary strings in the range $[\frac{2^k(j-1)}{n_i-1}, \frac{2^k(j-1)}{n_i-1} + R]$ , where $\Delta$ is a configurable constant value and the range $R < \frac{2^k}{2n_i-1}$. And, users with attribute value $a_j^{(i)}$ choose any one of the $p_j^{(i)} \cdot \Delta$ binary strings with equal probability $\frac{1}{p_j^{(i)}\Delta}$ as their mapping attribute value. Hence, after mapping, each attribute value is chosen with equal probability of $\frac{1}{\Delta}$. The big-jump mapping has three benefits: First, the entropy of the attribute $\mathcal{A}_i$ increases under the one-to-$N$ mapping. Second, different attributes are unified to the same measurement of $k$ bits. Third, even though the same attribute values are mapped then encrypted with different ciphertexts, the profile matching results will not change if the profiles are the Euclidean-distance close. We will show the correct matching rate under different Euclidean-distances in Section $IX$.

As an example, assume that the education social attributes of 100 users have equal weights, with four values (high school with probability 0.3, B.S. with probability 0.4, M.S. with probability 0.2, Ph.D. with probability 0.1). To increase the entropy of the education attribute, the users with attribute value 'high school' randomly choose one of $0.3\Delta$ 64-bit binary strings from the range $[1, 6 \times 10^8]$, the users with attribute value 'B.S.' randomly choose one of $0.4\Delta$ 64-bit binary string from the range $[1.4 \times 10^9, 2 \times 10^9]$, and so on.

**Attribute Chaining.** A naive approach to communicate attribute data to the server is to encrypt each user's attribute with the OPE after increasing their entropy and then send them to the untrusted server. Since the encrypted attribute values have the order relationships, the untrusted server can compare the order relationships of each encrypted attribute value to obtain the profile matching results (e.g., kNN matching [35], MAX-distance matching [35]).

However, although it is feasible to utilize the entropy-increase technique above to increase the privacy of profile matching, the ciphertexts still face the landmark node problem as we mentioned in $IV$. After the big-jump mapping, the encoded attribute values will not have uniform distribution in the message space, and accordingly, neither will the corresponding ciphertexts. Thus, it makes easier for an untrusted server to detect a landmark and to eventually obtain the plaintext. Hence, after we increase the entropy of the attributes, they are chained (i.e., combined) separately in random order. The randomization is done to prevent an attacker from obtaining the position of a specific attribute in the chain. Otherwise, the attacker does not need to brute-force the entire attribute data chain, but the specific string bits to obtain the attribute data (which has smaller entropy than the entire data chain).

Note that we randomize the orders of profile attributes in the chain and then the final chain is encrypted by OPE on the mobile device before it is sent it to the server. The format of the data message sent from user, $U$, to server, $S$, is as follows:

$$u \to S : ID_u, h(K_{up}), E_{K_{up}}(\mathcal{A}'_1)|| \dots ||E_{K_{up}}(\mathcal{A}'_n) \quad (3)$$

where $E_{k_{up}}$ is the OPE with the user's profile key $K_{up}$, $\mathcal{A}'_1, \dots, \mathcal{A}'_n$ is the entropy-increased attributes with random order.

**Profile Matching.** In the profile matching step, the server receives a profile matching query from user $u$ and tries to find the matching profiles for the interested user $u$. For this, the server first filters the stored encrypted profiles based on $h(K_{up})$, then measures the distance between the stored encrypted profiles and the encrypted profile of the interested user $u$.

*Definition 4:* The distance of user $u$ and user $v$ in the profile matching process at the server is defined as follows:

$$d(u,v) = \sum_{i=1}^{d} O(\mathcal{A}'^{(u)}_i) - \sum_{i=1}^{d} O(\mathcal{A}'^{(v)}_i) \quad (4)$$

where $O(\mathcal{A}'^{(u)}_i)$ is the order of user $u$'s profile attributes $\mathcal{A}'^{(u)}_i$.

With the comparable attribute values, the untrusted server can process profile matching using any matching algorithm (e.g., kNN matching and MAX-distance matching). For instance, assume the users have two attributes: Gender and Education. The message space is 60. Each user has a random-order attribute chain. User A (Gender=0, Education=1) has attribute chain 12|8, user B (Gender=0, Education=2) has attribute chain 34|2, and user C (Gender=1, Education=3) has attribute chain 50|48. Then, the order relationship in the server among users is as follows: user A has order 20 in total, user B has order 36 in total, and user C has order 98 in total. User A's profile is matched with that of user B.

**Profile Verification.** Assume that user $u$ wishes to verify whether user $v$'s identity and profile, which was matched by the server, is similar to hers. In other words, the profile is exactly matching under the same property-preserving key and exactly from the user the query results shows. However, user $v$ does not want to reveal her profile attribute values. The intuition behind our verification scheme is based on the *reversed fuzzy-commitment* [36], where a commitment can be opened using a set of *witnesses*, which is close to the original encoded witness.

For profile verification, each user $u$ owns a secret value $s_u$ generated by a random value generator. $h : G \to (0,1)^*$ is a one-way hash function and $p$ is a generator of the cyclic group $G$. To generate the authentication information $ciph_v$, user $v$ utilizes the profile key $K_{vp}$ based on the profile $A_v$ to encrypt the secret value, i.e., $ciph_v = E_{K_{vp}}(p^{s_v}||h(p^{s_u \cdot ID_v}))$, where $E()$ is a symmetric encryption scheme (e.g., AES-256). During the verification, user $u$, whose profile is close (i.e., $\| A_u - A_v \| \leq \theta$, where $\theta$ is the threshold of the RS Decoder) to that of user $v$, is able to decrypt user $v$'s authentication information $ciph_v = ciph_v^{(1)}||ciph_v^{(2)}$ by her profile key $K_{up}$. After obtaining the secret value $p^{s_v}$, user $v$ verifies whether the equation $h((ciph_v^{(1)})^{ID_v}) = ciph_v^{(2)}$ holds. If yes, the profile matching result, which states that user $v$ is a match, is considered as trustworthy.

For instance, assume the users have four attributes: Gender, Education, Interest 1 and Interest 2. User A has attribute chain 1|1|1|1, user B has attribute chain 2|2|2|3, and user C has attribute chain 2|3|3|2. Let the threshold $\theta$ of the RS decoder be 1, i.e., user B and user C with close Euclidean distance profiles will generate the same profile key $k_{p1}$ while user A will generate a different profile key $k_{p2}$. When user B obtains the authentication information $ciph_A$ and $ciph_C$, she can decrypt $ciph_C$ by her profile key $k_{p1}$ and obtain $t = p^{s_C}||h(p^{s_C \cdot ID_C})$. User B checks $(p^{s_C})^{ID_C}$ equal to the latter part of $ciph_C$ after hashing. In that case, the authentication information of user C is decrypted by user B, which will not be done in the authentication information of user A, because $p^{s_A}||h(p^{s_A \cdot ID_A})$ is encrypted by a different profile key $k_{p2}$ and cannot be decrypted by user B's profile key $k_{p1}$ .

## VII. Security analysis and performance

In this section, we first present the syntax of our scheme S-MATCH. Then we show the secruity analysis of S-MATCH by formally defining two attacks: the plaintext recovery under ordered known plaintext attack and the plaintext recovery under known key attack. Finally, cost analysis is given.

### A. Syntax and Correctness

*Definition 5:* S-MATCH is a privacy-preserving profile matching scheme for mobile social networks, which consists of a tuple $S - MATCH = (Keygen, InitData, Enc, Match, Auth, Vf)$.

- ***Key generation:*** $K_{up} \leftarrow Keygen(A_u)$. Keygen runs at the client's mobile device. The user's profile data $A_u$ is obtained from the profile interfaces of current social networks providers.
- ***Initialize data:*** $M_u \leftarrow InitData(A_u)$. InitData runs on the client's mobile device, which takes the user's raw

```
**Algorithm** $Keygen(A_u)$
1. $T^{(u)} \leftarrow RSD(A_u, \theta)$
2. $K' \leftarrow H(T^{(u)})$
3. return $K_{up} \leftarrow RSA - OPRF(K')$

**Algorithm** $InitData(A_u)$
1. Initialize the social profile data by increase the entropy of social profile dataset.
2. Chain the social profile data together with random order.
3. return $M_u$

**Algorithm** $Enc(M_u)$
1. return $OPE.Enc(K_{up}, M_u)$

**Algorithm** $Match(v, C)$
1. $C' \leftarrow EXTRA(h(K_{vp}), C)$
2. $C' \leftarrow SORT(C')$
3. $pos \leftarrow FIND(v, C')$
4. return $\{pos - \frac{k}{2},\ pos - 1, ...,\ pos + 1,\ pos + \frac{k}{2}\}$

**Algorithm** $Auth(u)$
1. return $AES.Enc(K_{up}, p^{s_u}||h(p^{s_u \cdot ID_u}))$

**Algorithm** $Vf(ID_v, ciph_v, u)$
1. $t \leftarrow AES.Dec(K_{up}, ciph_v)$
2. parse $t$ as two parts $t_1||t_2$
2. $s' \leftarrow h(t_1^{ID_v})$
3. $if(s' == t_2)$ then $b \leftarrow 1$
4. else $b \leftarrow 0$
5. return $b$
```

Fig. 3. The S-MATCH scheme.

profile data $A_u$ as input and outputs the newly computed high entropy profile data.

- ***Encryption:*** $C_u \leftarrow Enc(M_u)$. Enc runs OPE, the symmetric encryption scheme, on the client's mobile device.
- ***Matching:*** $R \leftarrow Match(u, C)$. Match runs at the server side to output $k$ users with similar social profiles to user $u$, with user $u$'s ID and the encrypted social profile dataset as input.
- ***Authentication:*** $ciph_u \leftarrow Auth(u)$. Auth runs at the the client's mobile device which takes user $u$'s ID as input and returns the authentication information $ciph_u$.
- ***Verification:*** $b \leftarrow Vf(ID_v, ciph_v, u)$. Vf runs at the the client's mobile device which takes the authentication information $ciph_v$ and user $v$'s ID as input and returns a boolean $b$.

## B. Security Analysis

Now, we formally define the plaintext recovery under ordered known plaintext attack (PR-OKPA) and the plaintext recovery under known key attack (PR-KK) which become more feasible given the information leakage problem and the key sharing problem we mentioned in Section IV.

*Definition 6:* PR-OKPA security game. The security game between a user $u$ and an adversary $S$ proceeds as follows:

(1) The user $u$ chooses the key $k \leftarrow K$.

(2) The user $u$ and the adversary $S$ engage in a polynomial number of rounds of interaction to recover plaintext $M$ of the ciphertext $C$. At round $i$,

- The adversary $S$ obtains the plaintext-ciphertext pairs $(M_i, C_i)$.
- The adversary leads the interaction for the ordered search in the server, with the adversary $S$ observing all the states of the server.

(3) The adversary $S$ outputs $M'$, its guess for the plaintext of ciphertext $C$.

We say the adversary $S$ wins the game if its guess is correct $(M' = M)$. Let $Adv_{S-MATCH}^{PR-OKPA}(S)$ be the value indicating the success of the adversary in the above game.

*Theorem 1:* A profile matching scheme based on OPE is PR-OKPA secure if for all adversaries S $Adv_{S-MATCH}^{PR-OCPA}(S) = \frac{\ln(2^e - 2) + 0.577}{2^{e-1}(2^e - 1)} \leq \frac{1}{2^\kappa}$, where $\kappa$ is the security level parameter, and $e$ is the entropy of the plaintext.

*Definition 7:* PR-KK security game. The security game between a user $u$ and an adversary $S$ proceeds as follows:

(1) The user $u$ colludes with an adversary $S$ by sharing his profile key $K_{up}$.

(2) The user $u$ and the adversary $S$ engage in a polynomial number of rounds of interaction to recover plaintext $M$ of the ciphertext $C$. At round $i$,

- The adversary $S$ hashes the $K_{up}$ as the index.
- The adversary leads the interaction for the ordered search in the server, with the adversary $S$ observing all the states of the server.
- The server returns the ciphertexts $\{C\}$.

(3) The adversary $S$ outputs $\{M'\}$.

We say the adversary $S$ wins the game if its guess is correct $(\{M'\} = \{M\})$. Let $Adv_{S-MATCH}^{PR-KK}(S)$ be the value indicating the success of the adversary in the above game.

*Theorem 2:* A profile matching scheme based on OPE is PR-KK secure if for all adversaries S $Adv_{S-MATCH}^{PR-KK}(S) = \frac{m}{N}$, where $m$ is the number of users whose profiles are close (i.e., $\| A_u - A_v \| \leq \theta$) to user $u$, $N$ is the number of users and $m \ll N$.

For the honest-but-curious server and honest-but-curious user (Section V-B), during the profile matching, only the order-preserving encrypted profiles will be submitted to the server. In other words, the server obtains nothing but the order of the plaintexts. With the entropy of users' profile attributes increased to a configurable value based on the security level, the scheme is PR-OKPA secure. For instance, to achieve the security level of 80, the entropy can be configured to 64 bits. Namely, even though the untrusted server obtains the order of the plaintext encrypted with OPE, it cannot obtain the exact values of users' profile attributes. Hence, our profile matching scheme is PR-OKPA secure against the honest-but-curious server.

Also, an honest-but-curious user is not able to obtain the profiles or network information of others because they do not communicate with each other directly. Additionally, the communication messages between the user and the server are protected from eavesdropping and modification by other users, because of the secure communication channel. During the results verification, the authentication information cannot be cracked by the honest-but-curious server or honest-but-curious user, because the verification scheme utilizes the user's secret value $s_u$ and the profile key $K_{up}$ to generate the authentication information. Assuming that the secret value $s_u$ is generated randomly and uniformly, the complexity to obtain $s_u$ from the $ciph_u$ is as hard as the computational Diffie-Hellman problem, which, as far as we know, cannot be solved in polynomial time in the proper group (e.g., the subgroup of quadratic residues). Consider the collusion between a user and the server, S-MATCH is PR-KK secure, which indicates that the profile leakage only happens among the users whose profiles are close (i.e., $\parallel A_u - A_v \parallel \leq \theta$) to this honest-but-curious user.

For the malicious server, it is impossible for the server to fake the profile matching results. This is because in the verification technique we proposed, each user is required to submit the authentication information $ciph_v$ for the profile verification. If the server wants to alter $ciph_v$ to fake the profile matching result, the server must obtain the user's profile key, which is impossible except for the collusion with the users. Also, the hash function $h()$ is assumed to be one-way and collision resistant, so the malicious server cannot reverse it to obtain the secret value. Therefore, our technique defends against an honest but curious server, honest but curious user, as well as a malicious server (that alters profile matching results).

### C. Cost Analysis

(1) *Computation cost*: For the mobile user, it takes $O(d)$ operations to increase the entropy and chain the attributes, $O(MN)$ operations for OPE, where $M$ is the plaintext length and $N$ is the ciphertext length. $d+2$ hash operations and 2 modular exponentiations are for profile key generation. One symmetric encryption operation and one symmetric decryption operation are needed for the verification protocol with profile key $k_{up}$. For the server, it takes $O(|V|log|V|)$ operations for sorting the users profile information, and $O(log|V|)$ operation for searching, where $|V|$ is the number of users under the same profile key.

(2) *Communication cost*: Communication is only between the server and the users. Users do not exchange information among themselves. To bootstrap the profile matching, users send their high-entropy profile information along with the authentication information, i.e., $l_{id} + l_h + l_{ciph} + d \cdot \frac{nN}{M}$, where $l_{id}$ is the length of the user ID, $l_h$ is the length of the hashed profile key as index, $l_{ciph}$ is the length of the authentication information, $n$ is the plaintext size of the profile attribute and $d$ is the number of the profile attributes. After obtaining the query request and conducting the profile matching, the server return the $k$ profile matching results with the corresponding authentication information, i.e., $k(l_{id} + l_{ciph})$, where $k$ is the number of the profile matching results, $l_{id}$ is the length of the user ID and $l_{ciph}$ is the length of the authentication information.

## VIII. Implementation

In this section, we present the implementation of S-MATCH client on an Android phone and S-MATCH server on a PC.

***Framework.*** The S-MATCH client was implemented as an Android application on HTC Nexus One smartphone, and S-MATCH server was implemented on a PC. The HTC Nexus One smartphone has 1 GHz QSD8250 processors running Android 2.1 platform. The PC had two 3.10 GHz Intel Core i5-2400 processors running the Linux 3.5 kernel.

***Communication.*** The HTC Nexus One communicates to the PC over an 802.11n 53Mbps WiFi connection. The application instances set up a communication channel via SSL socket using JAVA libraries, and the packages are sent with the mode Encrypt-then-MAC.

***Key Generation.*** We implemented RS decoder proposed in [32]. We used RSA in the javax.crypto package to implement the RSA-OPRF scheme as an instance of the oblivious pseudo-random function.

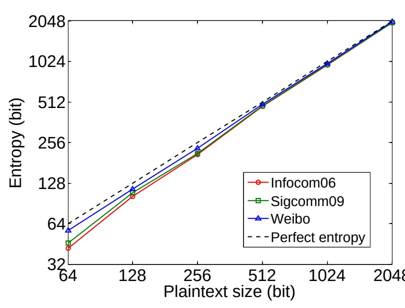
***Encryption.*** The OPE, as the instance of the PPE, was implemented in JAVA, based on the C++ code from [37]. As the ciphertexts are outsouced in the server, the ciphertext range in OPE is set as the same as the plaintext range. Based on the controlled trial in [8], the Paillier homomorphic cryptosystem is used as the instance of the homomorphic encryption,

***Verification.*** In the verification scheme, AES in CTR mode with random IV was utilized for symmetric encryption and decryption. The generator $p$ is generated from the BigInteger class in JAVA. SHA2 was used as the hash function.

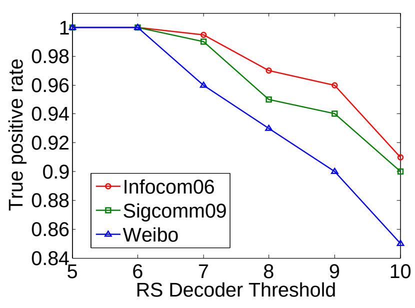
## IX. Performance Evaluation

In this section, we present an evaluation of the validity and efficiency of our scheme, S-MATCH, using three real-world datasets (Infocom06 [26], Sigcomm09 [27], and Weibo [28]) under a real testbed. In particular, we answer the following questions:

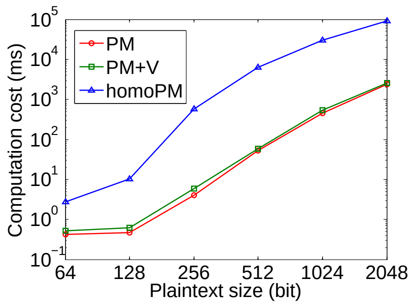
- *(information leakage)* How does our technique increase entropy for various plaintext sizes? How close is our technique to perfect entropy?
- *(correctness)* What is the true positive rate (correct matching rate) of the profile matching as the RS Decoder threshold varies?
- *(computation cost)* How do the client and the server computation costs of S-MATCH compare to a representative technique based on homomorphic encryption (homoPM [8]) in a real testbed?
- *(communication cost)* How dose the communication cost of S-MATCH compare to a representative technique based on homomorphic encryption (homoPM [8]) in a real testbed?



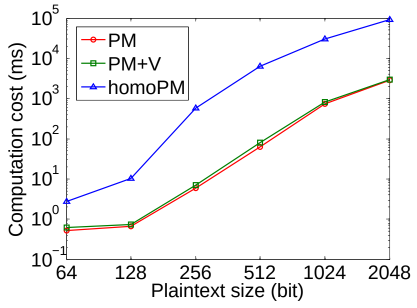
(a) Entropy of three datasets



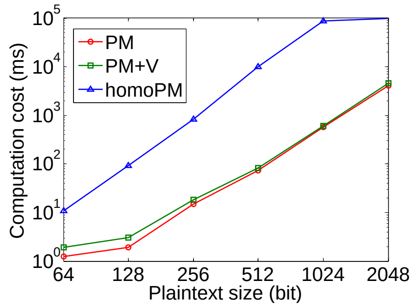
(b) True positive rate of the profile matching



(c) Computation cost of the client under Infocom06 dataset



(d) Computation cost of the client under Sigcomm09 dataset



(e) Computation cost of the client under Weibo dataset

Fig. 4. Entropy of the three datasets after applying our technique are indicated in Fig. 4(a). The true positive rate of the profile matching are indicated in Fig. 4(b). The user computation cost under three datasets are shown in Fig. 4(c), 4(d) and 4(e).

## A. *Information Leakage*

Figure $4(a)$ shows the entropy of the Infocom06, Sigcomm09 and Weibo datasets after the entropy increase and the attribute chaining, compared with that of the perfect entropy (theoretical limit). Overall, the entropy of the original data increases as the plaintext size $k$ increases. With the increase of the bit size for profile attributes, the entropy increasing step generates more one-to-$N$ mappings and $\log \Delta$ increases. Moreover, with the attribute chaining, the attribute value distribution among the message space becomes more uniform, which will also increase the entropy. Compared to the Infocom06 and Sigcomm09 datasets, the Weibo dataset has more profile attributes and users, which has larger original entropy. Similar to the Infocom06 and Sigcomm09 datasets, the increment of entropy after chaining becomes larger with the increase of the plaintext size $k$. As with more users and attributes, the increment of the entropy becomes slower when the plaintext size is small. However, when the plaintext size becomes larger, the rate of entropy increase grows faster. Therefore, the attribute data processed after entropy increase and attribute chaining is more suitable for PPE, which increases the security of PPE to protect from PR-OKPA.

## B. *Correctness*

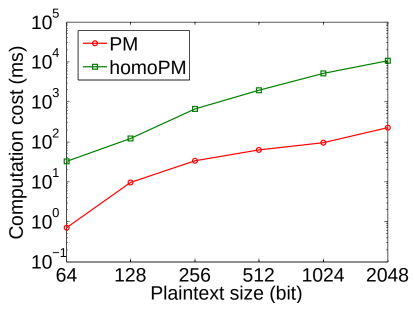
To evaluate the correctness of the profile matching of S-MATCH, we measured the true positive rate (TPR) of the profile matching results under various RS Decoder thresholds to generate the profile key. TPR indicates the proportion of true cases that are correctly found. The formal definition of TPR is as follows:

$$TPR = \frac{True\ Positive}{True\ Positive + False\ Negative}. \tag{5}$$

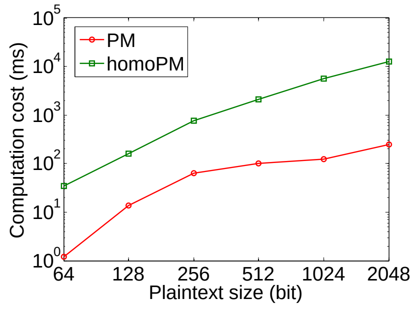
Figure $4(b)$ shows the TPR of the profile matching results under different RS Decoder thresholds in Infocom06, Sigcomm09 and Weibo datasets. The number of query results is set to 5, and the plaintext size is set to 64. For $\theta = 8$, our profile matching scheme has a correctness of $97.2\%$, $95.8\%$ and $93.0\%$ in the Infocom06 dataset, Sigcomm09 and Weibo dataset respectively. We can see that the TPR goes down as the RS Decoder threshold increases. This is caused by the correction threshold when implementing the RS decoding algorithm. For higher TPR, the Guruswami and Sudan algorithm [34] can be utilized to implement the RS decoding algorithm. Moreover, with the smaller RS Decoder threshold, the difference of the profiles under the same profile key is small, which helps the big-jump mapping in the entropy increase part have better performance. Another important observation is that the TPR of the profile matching decreases slightly in the Weibo dataset, which is caused by the fact that the Weibo dataset has more attributes which affect the performance of the big-jump mapping in the entropy increase part to distinguish the difference of the attributes.

## C. *Computation Cost Analysis*

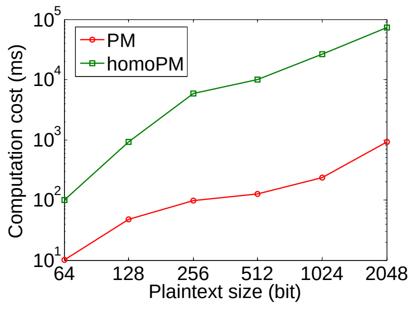
To evaluate the performance of our scheme, we measured the computation cost of S-MATCH under the Infocom06,



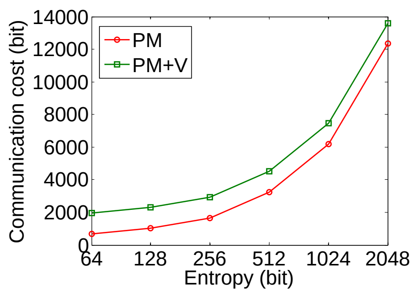
(a) Computation cost of the server under Infocom06 dataset



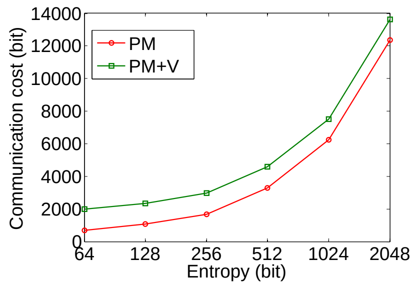
(b) Computation cost of the server under Sigcomm09 dataset



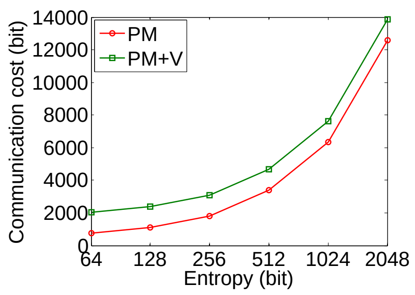
(c) Computation cost of the server under Weibo dataset



(d) Communication cost under Infocom06 dataset



(e) Communication cost under sigcomm09 dataset



(f) Communication cost under Weibo dataset

Fig. 5. The server computation cost under three datsets are shown in Fig. 5(a), Fig. 5(b) and Fig. 5(c). And Communication cost of S-MATCH under three datasets are shown in Fig. 4(c), Fig. 4(d) and Fig. 4(e).

Sigcomm09 and Weibo datasets in our mobile device and PC testbed. Figure 4(c), 4(d) and 4(e) present the client side computation cost of our scheme (i.e., PM and PM+V) in comparison to that of homoPM [8] under three real-world datasets. As the size of plaintext increases, the encryption time and data processing time of both schemes increase. In our profile matching scheme with the verification technique, when the plaintext size is small, the computation cost of the client side mainly comes from the key generation, which is relatively stable as the plaintext size increases. In the key generation, RS Decoder algorithm and RSA-OPRF algorithm are run, which take two modular exponentiations. As the plaintext size increases, the computation cost of the privacy-preserving profile matching scheme becomes larger, which is much less than that of the profile matching scheme based on homomorphic encryption. When the plaintext size is larger than 256 bits in the three social network datasets, the computation cost of our scheme is much less than that of those based on homomorphic encryption. With the plaintext size increases, the advantage of our scheme's lower computation cost becomes more obvious.

Figure 5(a), 5(b) and 5(c) presents the server computation cost of our scheme (i.e., PM) in comparison to that of homoPM [8] under three real-world datasets. Different from the PPE schemes, homomorphic encryption implemented by Paillier's cryptosystem [18] needs additional modular multiplication operations in the server after encryption to make the ciphertext comparable. In that case, the computation cost of homoPM includes an offline computation cost (i.e., encryption, which each user can process independently) and an online computation cost (i.e., a modular multiplication operation on the ciphertext, which makes the computation cost increase by the size of users and causes the latency in the query response). With a large real-world social network dataset and plaintext size, the profile matching based on PPE is more advantageous than homomorphic encryption. This is mainly due to the server computation cost of homomorphic encryption, where the computation cost of the multiplication increases with the increase of users and the attributes. Also, the attribute values encrypted by the Euclidean-distance close profiles key will further help to filter the dataset and reduce the search range, which also increases the performance.

## D. *Communication Cost Analysis*

To evaluate the performance of our scheme, we measured the communication cost of S-MATCH under the Infocom06, Sigcomm09 and Weibo datasets between the mobile device and PC.

Figure 5(d), Figure 5(e) and Figure 5(f) present the communication cost of our scheme (i.e., PM and PM+V). To evaluate the communication cost, the user ID length is defined as 32 bits, the number of the query results is 5, and $N = M$, where $M$ is the plaintext length and $N$ is the ciphertext length. In our scheme, the communication only happens between the untrusted server and the users. In other words, the packages to exchange the encrypted attribute values and the profile matching results is the main communication between them. The verification protocol introduces additional communication cost of the authentication information between the users and the server. As Figure 5(d), Figure 5(e) and Figure 5(f) indicate, the difference between the curve of PM and PM+V is the communication cost of the authentication information. Moreover, the communication cost under Weibo dataset is larger then that of Infocom06 dataset and Sigcomm09 dataset because of the larger number of the attributes. As the evaluation results indicate, our scheme is feasible for the low-bandwidth case.

## X. Conclusion and Future Work

In this paper, we presented a privacy-preserving profile matching scheme in large scale mobile social networks. Our scheme is based on property-preserving encryption (PPE). We proposed a key generation scheme for PPE to protect from the key-sharing problem when the user and the untrusted server collude. Also, we showed that entropy increase of social attribute data is necessary in PPE to achieve the privacy needed for the social profile matching process. For this, we evaluated the security of PPE over three real-world datasets. We analyzed the information leakage in the PPE schemes. A verification protocol was also proposed for users to verify the profile matching results, but learn nothing about other users' profile attributes. Finally, we showed that our scheme provided privacy and verification guarantees while achieving good performance in real-world social attribute datasets. The evaluation results indicated the efficiency and validity of our schemes. In our future work, we will further improve our scheme by increasing the efficiency of OPE. We plan to design our own OPE scheme which is able to choose the length of keys adaptively based on the entropy of social attributes.

## References

[1] A. Chaabane, G. Acs, M. Kaafar. You are what you like! Information leakage through users' interests In Proceedings of the 2012 NDSS.

[2] R. Popa, A. J. Blumberg, H. Balakrishnan, F. Li. Privacy and accountability for location-based aggregate statistics. In Proceedings of the 2011 ACM CCS, 653-666.

[3] C. Kaufma, R. Perlman, M. Speciner. Network security: private communication in a public world. Prentice Hall Press, 2002.

[4] P. Riley. The tolls of privacy: An underestimated roadblock for electronic toll collection usage. In Third International Conference on Legal, Security, and Privacy Issues in IT, 2008

[5] R. Reid. TomTom admits to sending your routes and speed information to the police, 2011. CNET UK

[6] M. Dijk, C. Gentry, S. Halevi, V. Vaikuntanathan Fully Homomorphic Encryption over the Integers. In Advances in Cryptology-Eurocrypt 2010, 24-43.

[7] J. Coron, A. Mandal, D. Naccache, M. Tibouchi Fully Homomorphic Encryption over the Integers with Shorter Public Keys. In Advances in Cryptology-Crypto 2011, 487-504.

[8] R. Zhang, Y. Zhang, J. Sun, and G. Yan. Fine-grained private matching for proximity-based mobile social networking. In Proceedings of the 2012 IEEE INFOCOM, 1969-1977.

[9] M. Li, N. Cao, S. Yu, W. Lou. FindU: Privacy-preserving personal profile matching in mobile social networks. In Proceedings of the 2011 IEEE INFOCOM, 2435-2443

[10] M. Arb, M. Bader, M. Kuhn, R. Wattenhofer. VENETA: Serverless Friend-of-Friend Detection in Mobile Social Networking. In Proceedings of the 2008 IEEE WIMOB, 184-189

[11] B. Wang, B. Li, H. Li. Gmatch: Secure and Privacy-Preserving Group Matching in Social Networks. In Proceedings of the 2012 IEEE GLOBECOM.

[12] M. Li, Z. Gao, S. Du. PriMatch: Fairness-aware Secure Friend Discovery Protocol in Mobile Social Network. In Proceedings of the 2012 IEEE GLOBECOM.

[13] O. Pandey, Y. Rouselakis. Property preserving symmetric encryption In Advances in Cryptology-EUROCRYPT 2012, 375-391.

[14] L. Zhang, X. Li, Y. Liu. Message in a sealed bottle: Privacy preserving friending in social networks. In Proceedings of the 2013 IEEE ICDCS.

[15] M. Nagy, N. Asokan, E. Cristofaro. Do I know you? - Efficient and Privacy-Preserving Common Friend-Finder Protocols and Applications. In Proceedings of the 2013 ACSAC.

[16] L. Kissner and D. Song. Privacy-preserving set operations. In Advances in Cryptology-Crypto 2005, 241-257.

[17] Q. Ye, H. Wang, and J. Pieprzyk. Distributed private matching and set operations. In Information Security Practice and Experience, 347-360: Springer, 2008.

[18] P. Paillier Public-key Cryptosystems based on Composite Degree Residuosity Classes. In Advances in cryptology-EUROCRYPT 1999, 223-238.

[19] R. Agrawal, J. Kiernan, R. Srikant, Y. Xu. Order preserving encryption for numeric data. In Proceedings of the 2004 ACM SIGMOD, 563-574.

[20] B. Alexandra. Order-preserving symmetric encryption. In Advances in Cryptology-EUROCRYPT 2009, 224-241: Springer, 2009.

[21] R. Popa, F. Li, N. Zeldovich. An Ideal-Security Protocol for Order-Preserving Encoding. In Proceedings of the 2013 S&P(Oakland).

[22] G. Ozsoyoglu, D. Singer, S. Chung. Anti-tamper databases: Querying encrypted databases. In Proceedings of 2003 Annual IFIP WG. 11: 4-6.

[23] W. Sun, B. Wang, N. Cao, M. Li, W. Lou, Y.T. Hou, and H. Li. privacy-preserving multi-keyword text search in the cloud supporting similarity-based ranking? In Proceedings of the 2013 ACM ASIACCS.

[24] J. Kornblum. Identifying almost identical files using context triggered piecewise hashing. Digit. Investig, 91-97. 2006.

[25] M. Srivatsa, M. Hicks. Deanonymizing mobility traces: using social network as a side-channel. In Proceedings of the 2012 ACM CCS, 628-637.

[26] J. Scott, R. Gass, J. Crowcroft, P. Hui, C. Diot, and A. Chaintreau. CRAWDAD dataset cambridge/haggle. http://crawdad.cs.dartmouth.edu/cambridge/haggle. [2009/05/29].

[27] J. Scott, R. Gass, J. Crowcroft, P. Hui, C. Diot, and A. Chaintreau. CRAWDAD dataset thlab/sigcomm2009. http://crawdad.cs.dartmouth.edu/thlab/sigcomm2009. [2012/07/15].

[28] WEIBO API. http://open.weibo.com/wiki/. [2013/07/10].

[29] E. Cho , S. Myers and J. Leskovec. Friendship and mobility: user movement in location-based social networks. In Proceedings of the 2011 ACM SIGKDD, 1082-1090.

[30] J. He, W Chu, Z Liu. Inferring privacy information from social networks. Intelligence and Security Informatics. Springer Berlin Heidelberg, 2006. 154-165.

[31] A. Traud, P. Mucha, M. Porter. Social structure of Facebook networks. Physica A: Statistical Mechanics and its Applications, 2012, 391(16): 4165-4180.

[32] E.R. Berlekamp. Algebriac Coding Theory. McGraw-Hill New York, 1968.

[33] J.L. Massey. Shift register synthesis and BCH decoding. Information Theory, IEEE Transactions on 15, no. 1 (1969): 122-127.

[34] V. Guruswami and M. Sudan. Improved decoding of reed-solomon and algebraic-geometric codes. In Proceedings of the 1988 IEEE FOCS, 28-37.

[35] T. Hastie, R. Tibshirani. The elements of statistical learning. New York: Springer, 2001.

[36] J. Katz, Y. Lindell. Introduction to modern cryptography. Chapman&Hall, 2008.

[37] R. Popa, C. Redfield, N. Zeldovich, and H. Balakrishnan. CryptDB: Protecting Confidentiality with Encrypted Query Processing. In Proceedings of the 2011 ACM SOSP. 85-100.