Privacy-preserving user profile matching in social networks

Yi, Xun; Bertino, Elisa; Rao, Fang-Yu; Lam, Kwok-Yan; Nepal, Surya; Bouguettaya, Athman

2019

Yi, X., Bertino, E., Rao, F.-Y., Lam, K.-Y., Nepal, S., & Bouguettaya, A. (2019). Privacy-preserving user profile matching in social networks. IEEE Transactions on Knowledge and Data Engineering, 32(8), 1572-1585. doi:10.1109/TKDE.2019.2912748

https://hdl.handle.net/10356/143529

https://doi.org/10.1109/TKDE.2019.2912748

Privacy-Preserving User Profile Matching in Social Networks

Xun Yi, Elisa Bertino, Fang-Yu Rao, Kwok-Yan Lam, Surya Nepal and Athman Bouguettaya

Abstract—In this paper, we consider a scenario where a user queries a user profile database, maintained by a social networking service provider, to identify users whose profiles are similar to the profile specified by the querying user. A typical example of this application is online dating. Most recently, an online data site, Ashley Madison, was hacked, which results in disclosure of a large number of dating user profiles. This data breach has urged researchers to explore practical privacy protection for user profiles in a social network. In this paper, we propose a privacy-preserving solution for profile matching in social networks by using multiple servers. Our solution is built on homomorphic encryption and allows a user to find out matching users with the help of multiple servers without revealing to anyone the query and the queried user profiles in the clear. Our solution achieves user profile privacy and user query privacy as long as at least one of the multiple servers is honest. Our experiments demonstrate that our solution is practical.

Keywords—User profile matching, data privacy protection, ElGamal encryption, Paillier encryption, homomorphic encryption

1 Introduction

Matching two or more users with related interests is an important and general problem, applicable to a wide range of scenarios including job hunting, friend finding, and dating services. Existing on-line matching services require participants to trust a third party server with their preferences. The matching server has thus full knowledge of the users' preferences, which raises privacy issues, as the server may leak (either intentionally, or accidentally) users' profiles.

When signing up for an online matching service, a user creates a "profile" that others can browse. The user may be asked to reveal age, sex, education, profession, number of children, religion, geographic location, sexual proclivities, drinking behavior, hobbies, income, religion, ethnicity, drug use, home and work addresses, favorite places. Even after an account is canceled, most online matching sites may retain such information.

Users' personal information may be re-disclosed not only to prospective matches, but also to advertisers and, ultimately, to data aggregators who use the data for purposes unrelated to online matching and without customer consent. In addition, there are risks such as scammers, sexual predators, and reputational damage that come along with using online matching services.

Many online matching sites take shortcuts with respect to safeguarding the privacy and security of their customers. Often, they use counterintuitive "privacy" settings, and their data management systems have serious security flaws.

In July 2015, "The Impact Team" group stole user data from Ashley Madison, a commercial website billed as enabling extramarital affairs. The group then threatened to release users' names and personally identifying information if Ashley Madison was not immediately shut down. On 18 and 20 August 2015, the group leaked more than 25 gigabytes of company data, including user details. Because of the site's policy of not deleting users' personal information, including real names, home addresses, search history and credit card transaction records, many users feared being publicly shamed. On 24 August 2015, Toronto police announced that two unconfirmed suicides had been linked to that data breach.

Such a data breach has raised growing concerns amongst users on the dangers of giving out too much personal information. Users of these services also need to be aware of data theft. A main challenge is thus how to protect privacy of user profiles in social networks. So far, the best solution is through encryption, i.e., users encrypt their profiles before uploading them onto social networks. However, when user profiles are encrypted, it is challenging to match the users with the similar profiles.

In this paper, we consider a scenario where a user queries a user profile database, maintained by a social networking service provider, to find out some users whose profiles are similar to the profile specified by the querying user. A typical example of this application is online dating. We give a privacy-preserving solution for user profile matching in social networks by using multiple servers.

Our basic idea can be summarized as follows. Before uploading his/her profile to a social network, each user encrypts the profile by a homomorphic encryption scheme with the common encryption key. Therefore, even if the user profile database falls into the hand of a hacker, the hacker can only get the encrypted data. When a user wishes to find people

X. Yi is with the School of Computer Science and Software Engineering, RMIT University, Melbourne, VIC 3001, Australia.
E. Bertino and F. Y. Rao are with the Department of Computer Science and Cyber Center, Purdue University, West Lafayette, IN 47907.
K. Y. Lam is with the School of Computer Science and Engineering, Nanyang Technological University, Singapore 639798
S. Nepal is with the Commonwealth Scientific and Industrial Research Organisation (CSIRO), Armidale, NSW 2350, Australia
A. Bouguettaya is with the School of Information Technologies, The University of Sydney, NSW 2006, Australia
Manuscript received 2 August 2017.

in the social network, the user encrypts his/her preferred user profile and a dissimilarity threshold and submits the query to the social networking service provider. Based on the query, multiple servers, which secretly share the decryption key, compare the preferred user profile with each record in the database. If the dissimilarity is less than the threshold, the matching user' contact information is returned to the querying user.

Our main contributions include

1) We formally define the user profile matching model, the user profile privacy and the user query privacy.
2) We give a solution for privacy-preserving user profile matching for a single dissimilarity threshold and then extend it for multiple dissimilarity thresholds.
3) We perform security analysis on our protocols. If at least one of multiple servers is honest, our protocols achieve user profile privacy and user query privacy.
4) We conduct extensive experiments on a real dataset to evaluate the performance of our proposed protocols under different parameter settings. Experiments show that our solutions are practical and efficient.

This paper extends our previous work [32] as follows.

1) In our previous work, we use a variant ElGamal encryption scheme [33] as the underlying homomorphic encryption scheme, which assumes the two prime factors of the modulus are public parameters. Rao [22] has found a security flaw in the encryption scheme, that is, an attacker may decrypt the ciphertexts without the decryption key. In this paper, we fix the security flaw by keeping the factorization of the modulus secret.
2) In our previous work, the user profile data is shared by all matching servers and therefore each matching server is required to maintain a user profile database. In this paper, we keep the user profile data in the social service provider only and therefore each matching server does not need to maintain any user profile database.
3) In our previous work, a query user can specify only one dissimilarity threshold for user matching. In this paper, we allow the query user to specify multiple dissimilarity thresholds for user matching.
4) In our previous work, user profile matching is based on numerical attributes only. In this paper, we extend our solution to user profile matching with categorical attributes.
5) In this paper, we enhance the security and performance of the private sharing and multiplication algorithms in our previous work. In addition, we provide two new collaborative algorithms for encryption and decryption.

The rest of the paper is organized as: Section 2 surveys related work. Section 3 introduces preliminaries. Section 4 describes our model for privacy-preserving user profile matching. Sections 5 and 6 present our solutions. Sections 7 and 8 present the security and performance analysis. The last section concludes the paper.

2 Related Work

In 2003, Agrawal et al. [1] gave a solution for private two-party set-intersection problem, where if one party P_1 inputs $X = \{x_1, x_2, \cdots, x_n\}$ and the other party P_2 inputs $Y = \{y_1, y_2, \cdots, y_n\}$, one party learns $X \cap Y$ and nothing else and the other party learns nothing. Their solution is based on commutative encryption with the property: $E_{k_1}(E_{k_2}(x)) = E_{k_2}(E_{k_1}(x))$, where k_1, k_2 are known to P_1 and P_2, respectively. The idea is: P_1 (P_2) encrypts its inputs X (Y) with its key k_1 (k_2), denoted as $E_{k_1}(X)$ ($E_{k_2}(Y)$) and exchanges them. Next, P_1 sends a pair $(E_{k_2}(Y), E_{k_1}(E_{k_2}(Y)))$ to P_2, which computes $E_{k_2}(E_{k_1}(X)))$, compares it with $(E_{k_2}(Y), E_{k_1}(E_{k_2}(Y)))$ and decrypts $E_{k_2}(y)$ if $E_{k_2}(E_{k_1}(x)) = E_{k_1}(E_{k_2}(y))$. Then Vaidya et al. [28] extended such a solution to n-party setting. Arb et al. [2] applied this idea to detect friend-of-friend in MSN.

In 2004, Freedman et al. [9] gave a solution for private two-party set-intersection problem based on polynomial evaluation. The idea is: P_1 defines a polynomial $P(y) = (x_1 - y)(x_2 - y)...(x_n - y)$ and sends to P_2 homomorphic encryptions of the coefficients of this polynomial. P_2 uses the homomorphic properties of the encryption system to evaluate the polynomial at each of his inputs and then multiplies each result by a fresh random number r to get an intermediate result, and adds to it an encryption of the value of his input, i.e., P_2 computes $\mathsf{E}(rP(y) + y)$. Therefore, for each of the elements in the intersection of the two parties' inputs, the result of this computation is the value of the corresponding element, whereas for all other values the result is random. In 2005, Kissner and Song [19] gave an improved solution that enables set-intersection, cardinality set-intersection, and over-threshold set-union operations on multisets. Later, Sang et al.[23], Ye et al. [31] and Dachman-Soled et al. [7] further improved and extended these solutions. In 2007, Li and Wu [17] proposed an unconditionally secure protocol for multi-party set intersection. Their idea is similar to Kissner and Song [19], while the inputs are shared among all parties using secret sharing [25], and computations are executed on those shares. In 2010, Narayanan et al. [20] proposed an improved solution. All these solutions are based on polynomial evaluation.

In 2008, Hazay and Lindell [13] proposed a solution for the private two-party set-intersection problem, where one party P_1 with a set X inputs the key k to a pseudorandom function F and another party P_2 with a set Y inputs the elements of its set. At the end, P_2 holds the set $F_k(y)_{y \in Y}$ while P_1 has learned nothing. Then, P_1 just needs to locally compute the set $F_k(x)_{x \in X}$ and send it to P_2. By comparing which elements appear in both sets, P_2 can learn the intersection (but nothing more). Their solution needs to compute a large number of exponentiations. In 2009, Jarecki and Liu [16] proposed a solution to improve the efficiency. In 2010, Cristofaro and Tsudik [6] proposed a solution based on blind RSA signatures [5]. The server computes $(H(x_j))^d$ for the client obliviously, where x_j's are client's inputs and d is server's one-time RSA signing key. Their solution is more efficient than previous solutions. All these solutions are based on pseudorandom functions.

In 2010, Yang et al. [29] proposed the E-SmallTalker scheme for matching people's interests before initiating a small-talk. E-SmallTalker considers n potential communication users $U_1, U_2, \cdots, U_n$, each having a set of interests $A_i =$

$\{a_{i,1}, a_{i,2}, \cdots, a_{i,n_i}\}$. It allows each user U_i to discover the identical interests in both A_i and A_j ($1 \leq i, j \leq n, i \neq j$) and the corresponding user U_j if there are identical interests between A_i and A_j. E-SmallTalker is based on Bloom filter [3]. An empty Bloom filter is a bit array of m bits, all set to 0. There must also be k different hash functions defined, each of which maps or hashes some set element to one of the m array positions with a uniformly random distribution. To add an element, one feeds it to each of the k hash functions to get k array positions and sets the bits at all these positions to 1. To query for an element (test whether it is in the set), one has to feed it to each of the k hash functions to get k array positions. If any of the bits at these positions is 0, the element is definitely not in the set - if it were, then all the bits would have been set to 1 when it was inserted. If all are 1, the element is in the set with certain probability. In 2011, Li et al. [18] proposed the FindU scheme for profile matching in mobile social networks. FindU is based on private set-intersection techniques. However, FindU is more efficient because it uses secure multi-party computation based on polynomial secret sharing [25].

In 2011, Huang et al. [15] proposed a privacy-preserving solution for biometric matching, where the server holds a database $\{v_i, p_i\}_{i=1}^n$ (v_i denotes the biometric data corresponding to some identity profile p_i), and the client who holds a biometric reading v' wants to learn the identity p_j for which v_j is the closets match to v' with respect to some metric (e.g., Euclidean distance), assuming this match is within some distance threshold δ. The idea is using Yao's garbled circuit technique [30] to perform the matching. Each gate of a circuit is associated with four ciphertexts (a garbled table) by one party. The collection of garbled tables (the garbled circuit) is sent to another party, who uses information obtained by oblivious transfer to learn the output of the function on the parties' inputs. This work is most related to our work. The major difference is that we assume the database is encrypted while they assume the database to be in cleartext.

In 2012, Shahandashti et al. [24] proposed a private fingerprint matching protocol that compares two fingerprints. The protocol enables two parties, each holding a private fingerprint, to find out if their fingerprints belong to the same individual. The main building block of their construction is the aided computation. Consider a polynomial $P(x) = \sum a_k x^k$ and a homomorphic encryption algorithm E for which Alice knows the decryption key. It is known that given $\{E(a_k)\}_{k=1}^n$ and x, the homomorphic property of E enables Bob to compute $E(P(x))$. Aided computation, on the other hand, enables Bob to compute $E(P(x))$ given $\{a_k\}_{k=1}^n$ and $E(x)$. A simplified description of their protocol is as follows. Let $\{p_i\}_{i=1}^n$ and $\{p'_j\}_{j=1}^m$ denote Alice's and Bob's fingerprint minutiae, respectively. They use the properties of homomorphic encryption schemes to privately calculate and compare the Euclidean distance and angular difference for each pair of minutiae (p_i, p'_j) to given thresholds. Their protocol enables Bob to calculate $E(z_{ij})$ for each pair (p_i, p'_j), such that z_{ij} is zero if the two minutiae match and non-zero otherwise. Then, using the aided computation idea, Bob calculates $E(R(z_{ij}))$, where R is a polynomial that maps zero to one and non-zero values to zero. Let $\sigma = \sum R(z_{ij})$, where σ equals the total number of minutia matchings. Using the homomorphic property of the encryption scheme $E(\sigma)$ can be calculated by Bob. Finally, the homomorphic property can be used to finalize the protocol and let Alice find out if σ is greater than or equal to a threshold τ or not. This work is also related to our work. The difference is that we assume the stored user data is encrypted while they assume two parties, each holding a private fingerprint in cleartext.

Most recently, Sun et al. [27] proposed privacy-preserving distance-aware encoding mechanisms to compare numerical values in the anonymous space. This work is also related to our work. The difference is that we encrypt the original data while they embed the original data into an anonymous space.

Our work is also closely related to homomorphic encryption, a form of encryption that allows computation on ciphertexts, generating an encrypted result which, when decrypted, matches the result of the operations as if they had been performed on the plaintext. The purpose of homomorphic encryption is to allow computation on encrypted data. Typical homomorphic encryption schemes include the ElGamal [8], Goldwasser-Micali [11], Paillier [21] schemes, which support either addition or multiplication on encrypted data, and the Boneh-Goh-Nissim [4] scheme, which supports arbitrary additions and one multiplication (followed by arbitrary additions) on encrypted data. Fully homomorphic encryption schemes, e.g., the Gentry scheme [10], supports arbitrary computation on encrypted data, but has not become practical so far.

3 Preliminaries

3.1 ElGamal Encryption

Our protocol is based on the ElGamal encryption scheme, which is introduced in this section.

The ElGamal encryption scheme consists of key generation, encryption, and decryption algorithms as follows.

- Key Generation: On input a security parameter k, it publishes a multiplicative cyclic group $\mathbb{G}$ of prime order q with a generator g, such that discrete logarithm problem over the group $\mathbb{G}$ is hard. Then it chooses a private key x randomly from $\mathbb{Z}_q^* = \{1, 2, \cdots, q-1\}$ and computes a public key $y = g^x$. The private key x is kept secret while the public key y can be known to everyone.
- Encryption: On inputs a message $m \in \mathbb{G}$ and the public key y, it chooses an integer r randomly from $\mathbb{Z}_q^*$ and outputs a ciphertext $C = \mathsf{E}(m) = (A, B)$, where $A = g^r$ and $B = m \cdot y^r$.
- Decryption: On inputs a ciphertext (A, B), and the private key x, it outputs the plaintext $m = \mathsf{D}(C) = B/A^x$.

3.2 Conditional Gate

A conditional gate [26] allows n (≥ 2) parties to multiply two encrypted values a and b without disclosing a and b, as long as a is restricted to a two-valued domain. It can be realized by the distributed ElGamal cryptosystem.

Assume the same setting as distributed ElGamal cryptosystem [32]. Let $\mathsf{E}(g^a), \mathsf{E}(g^b)$ denote ElGamal encryptions, with $a \in$

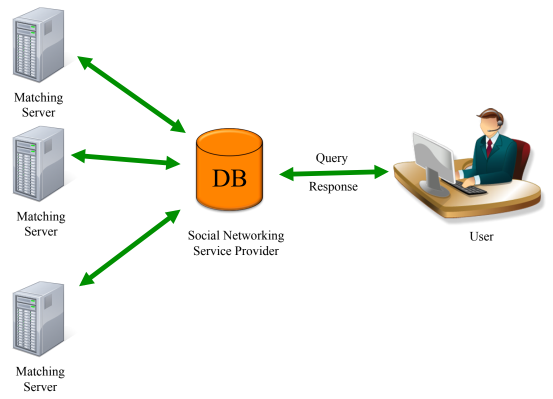


Fig. 1: Our Model for Privacy-Preserving User Profile Matching

$\{-1,1\}$ and $b \in \mathbb{Z}_q$, the n server jointly compute $W = \mathsf{E}(g^{ab})$ using the conditional gate as follows[1].

- Let $U_0 = \mathsf{E}(g^a)$ and $V_0 = \mathsf{E}(g^b)$
- For $i = 1, 2 \cdots, n$, the server S_i takes U_{i-1} and V_{i-1} as input and outputs $U_i = \mathsf{RE}(U_{i-1}^{r_i})$ and $V_i = \mathsf{RE}(V_{i-1}^{r_i})$, where r_i is randomly chosen from $\{-1,1\}$ and RE stands for re-encryption.
- The n servers jointly decrypt U_n to obtain $g^{a\prod_{i=1}^n r_i}$. Because $a, r_i \in \{-1,1\}$, it is easy to know $a' = a\prod_{i=1}^n r_i \in \{-1,1\}$.
- Let $W = V_n^{a'}$.

The conditional gate correctly computes $\mathsf{E}(g^{ab})$ because

$$V_n^{a'} = \mathsf{E}(g^{b\prod_{i=1}^n r_i})^{a'} = \mathsf{E}(g^{ab(\prod_{i=1}^n r_i)^2}) = \mathsf{E}(g^{ab}).$$

Based on the conditional gate, the n servers can jointly compute $\mathsf{E}(g^{ab})$ given $\mathsf{E}(g^a)$ and $\mathsf{E}(g^b)$ for $a, b \in \{0,1\}$ as follows.

$$\mathsf{E}(g^{ab}) = \mathsf{E}(g^{((2a-1)b+b)2^{-1}}) = (\mathsf{E}(g^{(2a-1)b})\mathsf{E}(g^b))^{2^{-1}}.$$

Note that $2a-1 \in \{-1,1\}$ when $a \in \{0,1\}$ and $\mathsf{E}(g^{(2a-1)b})$ can be computed with the conditional gate as above.

Furthermore, given encrypted bit representations $\mathsf{E}(x) = (\mathsf{E}(g^{x_{m-1}}), \cdots, \mathsf{E}(g^{x_1}), \mathsf{E}(g^{x_0}))$ and $\mathsf{E}(y) = (\mathsf{E}(g^{y_{m-1}}), \cdots, \mathsf{E}(g^{y_1}), \mathsf{E}(g^{y_0}))$ of two numbers x and y, the n servers can jointly compute encrypted bits of $x+y$, given by $\mathsf{E}(z) = \mathsf{E}(x+y) = (\mathsf{E}(g^{c_{m-1}}), \mathsf{E}(g^{z_{m-1}}), \cdots, \mathsf{E}(g^{z_1}), \mathsf{E}(g^{z_0}))$ as follows:

$$\mathsf{E}(g^{c_i}) = \mathsf{E}(g^{x_i y_i})\mathsf{E}(g^{x_i c_{i-1}})\mathsf{E}(g^{y_i c_{i-1}})\mathsf{E}(g^{-2x_i y_i c_{i-1}}),$$
$$\mathsf{E}(g^{z_i}) = \mathsf{E}(g^{x_i})\mathsf{E}(g^{y_i})\mathsf{E}(g^{c_{i-1}})\mathsf{E}(g^{-2c_i}),$$

for $i = 0, 1, \cdots, m-1$, where $c_{-1} = 0$. We denote $\mathsf{E}(x+y) = \mathsf{E}(x) \boxplus \mathsf{E}(y)$.

Yao [30] garbled circuit allows two parties, each having a private input bit, compute the product of their private inputs. However, it is restricted to two parties only, whereas the conditional gate allows more than two parties to compute the product of their private inputs.

1. The interested reader is referred to [26] for the detailed description of the protocol.

4 Model for Privacy-Preserving User Profile Matching

4.1 Our Model

Our model considers a social networking service environment with users and servers. Our model with one user, one database (DB) server and n matching servers is illustrated in Fig. 1.

In our model, all users stores their profiles in the DB server in the service provider. User profile attributes are either sensitive or insensitive. We consider protection of sensitive attributes only. In addition, user profile attributes are either numeric (e.g., income) or categorical (e.g., address). We consider numeric attributes only in this paper except in Section 6.2.

4.2 Dissimilarity Definitions

Definition 1 (Dissimilarity Definition for Single Attribute). For any numerical attribute A, the dissimilarity between two users U and U' with $A = a$ and $A = a'$, respectively, is defined as $d_A(U, U') = (a - a')^2$, where a, a' are integers.

Definition 2 (Dissimilarity Definition for Multiple Attributes). Assume that $A_1, A_2, \cdots, A_m$ are m numerical attributes for user profile, the dissimilarity between two users U and U' is defined as $d(U, U') = \sum_{i=0}^m d_{A_i}(U, U')$.

For a user who queries the user profile database, different attributes may have different impacts on the dissimilarity. Some attributes may be more important for the dissimilarity than other attributes. We introduce the concept of weight to measure the importance of an attribute to the dissimilarity. The weight for an attribute is an integer more than or equal to 0. It is specified by a user who queries the database.

Definition 3 (Weighted Dissimilarity Definition for Multiple Attributes). Assume that $A_1, A_2, \cdots, A_m$ are m numerical attributes for user profile and the weight of attribute A_i is w_i. The weighted dissimilarity between two users U and U' is defined as $d(U, U') = \sum_{i=0}^m w_i d_{A_i}(U, U')$.

4.3 Security Definitions

To protect the user profile privacy, the ElGamal encryption is used. At first, each matching server S_i generates its ElGamal public/private key pair (pk_i, sk_i). The common public key of the social networking service is set to $PK = \prod_{i=1}^n pk_i$.

According to the common public key PK, a user encrypts each attribute of his profile along with his contact information and sends the encrypted profile to the DB server, which stores them in a user profile database. The encrypted profile can be decrypted only by the cooperation of the n matching servers.

To find users with similar profiles, a user specifies a profile, encrypts and submits it along with a dissimilarity threshold to the social networking service provider.

The n matching servers cooperate to find the matching users from the user profile database according to the user profile and the dissimilarity threshold and then return the contact information of the matching users with dissimilarity less than the threshold to the querying user.

Considering m attributes $A_1, A_2, \cdots, A_m$ for user profile, we formally define the above process with a user profile matching protocol, composed of four algorithms as follows.

(1) Profile Generation (PG): Takes as input the common public key PK of the social networking service, the profile $a_1, a_2, \cdots, a_m$ of the user, (the user) outputs an encrypted profile P, denoted as $P = \mathsf{PG}(PK, a_1, a_2, \cdots, a_m)$. The encrypted profile P is submitted to the social networking service provider and stored in the user profile database DB.

(2) Query Generation (QG): Takes as input the common public key PK of the social networking service, the profile $a_1, a_2, \cdots, a_m$ specified by the user, the corresponding wights $w_1, w_2, \cdots, w_m$, and the dissimilarity threshold δ, (the user) outputs a query Q, denoted as $Q = \mathsf{QG}(PK, a_1, w_1, a_2, w_2 \cdots, a_m, w_m, \delta)$, which is submitted to the social networking service provider.

(3) Response Generation (RG): Takes as input the query Q, the user profile database DB, and the private keys $sk_1, sk_2, \cdots, sk_n$, (the n matching servers) outputs a response R, denoted as $R = \mathsf{RG}(Q, DB, sk_1, sk_2, \cdots, sk_n)$. The response R, containing the contact information of the first t matching users in DB, is then returned to the user, where t is either fixed or specified by the user in Q.

(4) Response Retrieval (RR): Takes as input the response R, (the user) outputs the first t matching users in DB.

A user profile matching protocol is correct if the response R lists the contact information of the first t' matching users with the dissimilarity less than the threshold δ, where $t' = min(t, T)$ and T denotes the total number of matching users in DB.

Now, we formally define user profile privacy with a game as follows.

Given the common public key PK, consider the following game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. The game consists of the following steps:

(1) The adversary $\mathcal{A}$ chooses two user profiles $(a_{01}, a_{02}, \cdots, a_{0m})$ and $(a_{11}, a_{12}, \cdots, a_{1m})$ and sends them to the challenger $\mathcal{C}$.

(2) The challenger $\mathcal{C}$ chooses a random bit $b \in \{0, 1\}$, and executes the Profile Generation (PG) to obtain an encrypted profile $P_b = \mathsf{PG}(PK, a_{b1}, a_{b2}, \cdots, a_{bm})$, and then sends P_b back to the adversary $\mathcal{A}$.

(3) The adversary $\mathcal{A}$ can experiment with the code of P_b in an arbitrary non-black-box way, and finally outputs a bit $b' \in \{0, 1\}$.

The adversary wins the game if $b' = b$ and loses otherwise. We define the adversary $\mathcal{A}$'s advantage in this game to be $\mathsf{Adv}_{\mathcal{A}}(k) = |\mathsf{Pr}(b' = b) - 1/2|$ where k is the security parameter.

Definition 4 (User Profile Privacy Definition). In a user profile matching protocol, the user has profile privacy if for any probabilistic polynomial time (PPT) adversary $\mathcal{A}$, we have that $\mathsf{Adv}_{\mathcal{A}}(k)$ is a negligible function, where the probability is taken over coin-tosses of the challenger and the adversary.

User profile privacy ensures that the attacker cannot determine the profile of the user even if some matching servers are compromised by the attacker.

Next, we formally define user query privacy with a game as follows: Given the common public key PK, consider the following game between an adversary $\mathcal{A}$, and a challenger $\mathcal{C}$. The game consists of the following steps:

(1) The adversary $\mathcal{A}$ chooses two user profiles $(a_{01}, a_{02}, \cdots, a_{0m})$ with corresponding weights $(w_{01}, w_{02}, \cdots, w_{0m})$ and a dissimilarity threshold δ_0, and $(a_{11}, a_{12}, \cdots, a_{1m})$ with corresponding weights $(w_{11}, w_{12}, \cdots, w_{1m})$ and another dissimilarity threshold δ_1, and sends them to the challenger $\mathcal{C}$.

(2) The challenger $\mathcal{C}$ chooses a random bit $b \in \{0, 1\}$, and executes the Query Generation (QG) to obtain a query $Q_b = \mathsf{QG}(PK, a_{b1}, w_{b1}, a_{b2}, w_{b2}, \cdots, a_{bm}, w_{bm}, \delta_b)$, and then sends Q_b back to the adversary $\mathcal{A}$.

(3) The adversary $\mathcal{A}$ can experiment with the code of Q_b in an arbitrary non-black-box way, and finally outputs a bit $b' \in \{0, 1\}$.

The adversary wins the game if $b' = b$ and loses otherwise. We define the adversary $\mathcal{A}$'s advantage in this game to be $\mathsf{Adv}_{\mathcal{A}}(k) = |\mathsf{Pr}(b' = b) - 1/2|$ where k is the security parameter.

Definition 5 (User Query Privacy Definition). In a user profile matching protocol, the user has user query privacy if for any probabilistic polynomial time (PPT) adversary $\mathcal{A}$, we have that $\mathsf{Adv}_{\mathcal{A}}(k)$ is a negligible function, where the probability is taken over coin-tosses of the challenger and the adversary.

User query privacy ensures that the attacker cannot determine the query of the user even if some matching servers are compromised by the attacker.

5 User Profile Matching Protocol

5.1 Initialization

We will use the ElGamal encryption scheme [8] and choose the public parameters on the basis of the Paillier encryption scheme [21].

On input a security parameter k, the system generates two large distinct primes p and q and computes $N = pq$ and $g = (1 + N)^p r^N (mod\ N^2)$ for a randomly chosen integer $r \in \{2, 3, \cdots, N^2 - 1\}$, such that $(g - 1)/N$ is not an integer, then publishes (N, g) and erase p and q from the memory. Note that (N, g) can be privately generated by multiple parties as described in [14] such that no one knows p and q if at least one party can be trusted, or by the Intel Software Guard Extensions (Intel SGX). Anyone can check if $(g - 1)/N$ is an integer.

Based on the public parameters (N, g), each matching server S_i chooses a private key SK_i randomly from $\mathbb{Z}_\rho^* = \{1, 2, \cdots, \rho - 1\}$, where ρ is a large integer less than N, and computes and publishes a public key $PK_i = g^{SK_i} (mod\ N^2)$ and a zero-knowledge proof of knowledge of SK_i. Let the common public key of the social networking service be $PK = \prod_{i=1}^n PK_i (mod\ N^2)$.

According to the common public key PK and the public parameters (N, g), each user U_i encrypts his profile $(a_{i1}, a_{i2}, \cdots, a_{im})$ as follows,

$$\mathsf{E}(g_1^{a_{ij}}) = (g^{r_{ij}} (mod\ N^2), g_1^{a_{ij}} PK^{r_{ij}} (mod\ N^2))$$

where r_{ij} is randomly chosen from $\mathbb{Z}_\rho^*$ and $g_1 = N + 1$. We assume that $|a_{ij}| \ll N$.

In the same way, the user U_i also encrypts $(a_{i1}^2, a_{i2}^2, \cdots, a_{im}^2)$. In addition, the user U_i encrypts his contact information CI_i, e.g., email address or mobile phone number, where we assume that the size of the binary representation of CI_i is less than the size of N, i.e., $CI_i < N$.

Then the user U_i submits

$$\mathsf{E}(g_1^{CI_i}), \mathsf{E}(g_1^{a_{i1}}), \mathsf{E}(g_1^{a_{i1}^2}), \cdots, \mathsf{E}(g_1^{a_{im}}), \mathsf{E}(g_1^{a_{im}^2})$$

to the social networking service provider, which stores them in the user profile database DB.

The database DB containing l users' profiles is a set

$$DB = \{\mathsf{E}(g_1^{CI_i}), [\mathsf{E}(g_1^{a_{ij}}), \mathsf{E}(g_1^{a_{ij}^2})]_{1\le j\le m} | 1 \le i \le l\}.$$

Since the user profiles are encrypted by the common public key PK, nobody knows the user profiles unless the n matching servers in the social network collude. But we assume that at least one out of the n servers is trusted not to collude with other servers to attack the user profiles. Therefore, the privacy of the user profiles can be protected.

5.2 Private User Profile Matching

When a user U wishes to find matching users from the social network, he specifies his preferred profile attributes $A_1 = a_1, A_2 = a_2, \cdots, A_m = a_m$, and the corresponding weights $w_1, w_2, \cdots, w_m$, and the dissimilarity threshold δ. We assume that all w_j are integers between 0 and $2^{\ell_w}-1$ and δ is an integer between 0 and $2^{\ell_\delta}-1$, where ℓ_w and ℓ_δ are the bit-length of w_j's and δ, respectively. In addition, based on the public parameters (N, g), the user U randomly chooses the private key SK_U from $\mathbb{Z}_\rho^*$ and computes the public key $PK_U = g^{SK}(mod\ N^2)$.

Then the user U generates a query

$$Q = \{\mathsf{E}(g_1^{a_1}), \mathsf{E}(g_1^{w_1}), \cdots, \mathsf{E}(g_1^{a_m}), \mathsf{E}(g_1^{w_m}), \mathsf{E}(g_1^{\delta}), PK_U\}$$

and submits Q to the social networking service provider.

After receiving the query Q from the user U, the n matching servers cooperate to find matching users from the database DB in four steps as follows.

Private Sharing of Query. Given an encryption of g_1^x, i.e., $\mathsf{E}(g_1^x)$, assume that the absolute value of x is smaller than L, where $L = 2^{\ell_x+\ell} \ll N$, ℓ_x is the bit-length of x, and ℓ is a sufficiently large statistic security parameter to mask x, e.g., the values of attributes, weights, and thresholds, etc., the n matching servers can cooperate to privately share x by running Algorithm 1.

Algorithm 1 Private Sharing (PS) (n Servers)

```
Input: $\mathsf{E}(g_1^x), N, g, PK$ (public), $S_1 : SK_1, S_2 : SK_2, \cdots, S_n, SK_n$ (private)
Output: $S_1 : x_1, \cdots, S_n : x_n$ such that $\sum_{k=1}^n x_i = x$.
1: let $(A, B) = \mathsf{E}(g_1^x)$
2: for $k = 1$ to $n-1$ {
3:     The server $S_k$ randomly chooses an integer $x_k$ from $(-L, L)$, computes $(g_1^{x_k} A^{SK_k})^{-1}(mod\ N^2)$ and sends it to the server $S_n$.
4: }
5: $S_n$ computes
       $x_n = [B \prod_{k=1}^{n-1} (g_1^{x_k} A^{SK_k})^{-1}/A^{SK_n}(mod\ N^2) - 1]/N$
6: return $S_1 : x_1, S_2 : x_2, \cdots, S_n : x_n$
```

Theorem 1 (Private Sharing Correctness). In Algorithm 1, we have $\sum_{k=1}^n x_k = x$.

Proof. Assume $A = g^r$ and $B = g_1^x PK^r$ for a random integer r, we have

$$\begin{aligned} & B\prod_{k=1}^{n-1}(g_1^{x_k}A^{SK_k})^{-1}/A^{SK_n} \\ = \;& g_1^{x-\sum_{k=1}^{n-1}x_k} PK^r / \prod_{k=1}^{n} g^{SK_k r} \\ = \;& g_1^{x-\sum_{k=1}^{n-1}x_k} = (1+N)^{x-\sum_{k=1}^{n-1}x_k} \\ = \;& 1 + (x - \sum_{k=1}^{n-1} x_k)N\ (mod\ N^2) \end{aligned}$$

$$[B\prod_{k=1}^{n-1}(g_1^{x_k}A^{SK_k})^{-1}/A^{SK_n}(mod\ N^2) - 1]/N = x - \sum_{k=1}^{n-1} x_k$$

i.e., $x_n = x - \sum_{k=1}^{n-1} x_k$. The theorem is proved. $\triangle$

After the executions of Algorithm 1 for a_j and w_j ($j = 1, 2, \cdots, m$), respectively, the server S_k ($k = 1, 2, \cdots, n$) will privately hold $a_j^{(k)}$ such that $\sum_{k=1}^n a_j^{(k)} = a_j$, and $w_j^{(k)}$ such that $\sum_{k=1}^n w_j^{(k)} = w_j$.

Private Computation of Dissimilarity. Assume that the server S_k ($k = 1, 2, \cdots, n$) privately holds x_j such that $\sum_{k=1}^n x_j = x$, given the encryption of g_1^y, i.e., $\mathsf{E}(g_1^y)$, the n matching servers can cooperate to privately compute $\mathsf{E}(g_1^{xy})$ by running Algorithm 2.

Algorithm 2 Private Multiplication (PM) (n servers)

```
Input: $S_1 : x_1, S_2 : x_2, \cdots, S_n : x_n$ (private), $\mathsf{E}(g_1^y)$, $N, g, PK$ (public)
Output: $\mathsf{E}(g^{xy})$.
1: for $k = 1$ to $n$ {
2:     $S_k$ computes and re-encrypts $\mathsf{E}(g_1^y)^{x_k} = \mathsf{E}(g_1^{x_k y})$
3:     $S_k$ broadcasts $\mathsf{E}(g_1^{x_k y})$ to other servers.
4:     $S_k$ computes $(A, B) = \mathsf{E}(g_1^{x_1 y})\mathsf{E}(g_1^{x_2 y}) \cdots \mathsf{E}(g_1^{x_n y})$
5: }
6: return $(A, B)$
```

Theorem 2 (Private Multiplication Correctness). In Algorithm 2, we have $(A, B) = \mathsf{E}(g_1^{xy})$.

Proof. Because

$$\begin{aligned} (A, B) &= \mathsf{E}(g_1^{x_1 y})\mathsf{E}(g_1^{x_2 y}) \cdots \mathsf{E}(g^{x_n y}) \\ &= \mathsf{E}(g_1^{(x_1+x_2+\cdots+x_n)y}) = \mathsf{E}(g_1^{xy}) \end{aligned}$$

The theorem is proved. $\triangle$

Now the n servers cooperate to determine the dissimilarity between the user profile and the given record $(\mathsf{E}(g_1^{a_{i1}}), \mathsf{E}(g^{a_{i1}^2}), \mathsf{E}(g^{a_{i2}}), \mathsf{E}(g^{a_{i2}^2}) \cdots, \mathsf{E}(g_1^{a_{im}}), \mathsf{E}(g_1^{a_{im}^2}))$ of a user U_i from the database DB.

Because the server S_k ($k = 1, 2, \cdots, n$) has privately held $a_j^{(k)}$ such that $\sum_{k=1}^n a_j^{(k)} = a_j$, the n servers can cooperate privately to compute $\mathsf{E}(g_1^{(a_j-2a_{ij})a_j})$ with Algorithm 2 and then $\mathsf{E}(g_1^{(a_j-2a_{ij})a_j})\mathsf{E}(g_1^{a_{ij}^2}) = \mathsf{E}(g_1^{(a_j-a_{ij})^2})$, where $j = 1.2, \cdots, m$.

Next, because the server S_k ($k = 1, 2, \cdots, n$) has privately held $w_j^{(k)}$ such that $\sum_{k=1}^n w_j^{(k)} = w_j$ ($j = 1.2, \cdots, m$), each server S_k can compute and broadcast

$$\prod_{j=1}^{m} \mathsf{E}(g_1^{w_j^{(k)}(a_j-a_{ij})^2}) = \mathsf{E}(g_1^{\sum_{j=1}^m w_j^{(k)}(a_j-a_{ij})^2})$$

At last, the n servers can compute

$$\prod_{k=1}^{n} \mathsf{E}(g_1^{\sum_{j=1}^m w_j^{(k)}(a_j-a_{ij})^2}) = \mathsf{E}(g_1^{\sum_{k=1}^n\sum_{j=1}^m w_j^{(k)}(a_j-a_{ij})^2})$$

$$= \mathsf{E}(g_1^{\sum_{j=1}^m w_j(a_j-a_{ij})^2}) = \mathsf{E}(g_1^{d(U,U_i)}).$$

Given the encryption of the threshold, i.e., $\mathsf{E}(g_1^{\delta})$, the n servers can compute $\mathsf{E}(g_1^{\delta})/\mathsf{E}(g_1^{d(U,U_i)}) = \mathsf{E}(g_1^{\delta-d(U,U_i)})$ and then privately share $d = \delta - d(U, U_i)$ by running Algorithm 1. After execution of Algorithm 1, assume that the server S_k ($k = 1, 2, \cdots, n$) privately holds d_j such that $\sum_{k=1}^n d_j = d$, where $|d_j| < L$ for $j = 1, 2, \cdots, n-1$, $|d_n| < nL$ and $|d| \ll L$.

Private Comparison with Threshold. For the purpose of security and efficiency, we will use the original ElGamal encryption in this step. The n servers cooperate to choose a multiplicative cyclic group $\mathbb{G}$ of prime order q' with a generator g_2, such that discrete logarithm problem over the group $\mathbb{G}$ is hard. Then each server S_k chooses a private key sk_k randomly from $\mathbb{Z}_{q'}^* = \{1, 2, \cdots, q'-1\}$ and computes a public key $pk_k = g_2^{sk_k}$. The common public key is defined as $pk = \prod_{k=1}^n pk_i$.

Let $\lambda = \lceil \log_2 nL \rceil + 2$, assume the two's complement of d_k is $\overline{d_k} = b_{\lambda-1}^{(k)} b_{\lambda-2}^{(k)} \cdots b_1^{(k)} b_0^{(k)}$ where $b_{\lambda-1}^{(k)} = 0$ if $d_k \geq 0$ and $b_{\lambda-1}^{(k)} = 1$ otherwise according to [12].

To compare $d(U, U_i)$ and δ, the n servers cooperate to run Algorithm 3, where e stands for the original ElGamal encryption with the public key pk.

Algorithm 3 Private Comparison (PC) (n servers)

```
Input: $S_1 : \overline{d_1}, S_2 : \overline{d_2}, \cdots, S_n : \overline{d_n}$ (private), $q'$, $g_2$, $\mathbb{G}$, $pk$ (public)
Output: $\mathsf{e}(1)$ if $d(U,U_i) \leq \delta$ and $\mathsf{e}(g_2)$ otherwise.
1: let $S = (\mathsf{e}(g_2^{s_{\lambda-1}}), \cdots, \mathsf{e}(g_2^{s_1}), \mathsf{e}(g_2^{s_0}))$ where $s_j = 0$ for all $j$.
2: for $k = 1$ to $n$ {
3:    $S_k$ encrypts $\overline{d_k}$ to get
         $$\mathsf{e}(\overline{d_k}) = (\mathsf{e}(g_2^{b_{\lambda-1}^{(k)}}), \cdots, \mathsf{e}(g_2^{b_1^{(k)}}), \mathsf{e}(g_2^{b_0^{(k)}}))$$
4:    $n$ servers cooperate to compute
         $$S = S \boxplus \mathsf{e}(\overline{d_k}) = (\mathsf{e}(g_2^{s_{\lambda-1}}), \cdots, \mathsf{e}(g_2^{s_1}), \mathsf{e}(g_2^{s_0}))$$
      as described in Section 3.2, where $s_j$ is either 0 or 1.
5: }
6: return $\mathsf{e}(g_2^{s_{\lambda-1}})$
```

Theorem 3 (Private Comparison Correctness). In Algorithm 3, we have $s_{\lambda-1} = 0$ if $d(U, U_i) \leq \delta$ and 1 otherwise.

Proof. According to the binary addition described in Section 3.2, we have $(s_{\lambda-1} \cdots s_1 s_0)$ is the two's complement of $d = \delta - d(U, U_i)$. Based on the properties of two's complement, we have $s_{\lambda-1} = 0$ if $d \geq 0$ (i.e., $d(U, U_i) \leq \delta$) and 1 otherwise. The theorem is proved. $\triangle$

Private Generation of Response. After obtaining $\mathsf{e}(g_2^{s_{\lambda-1}})$, the n servers cooperate to decrypt it by running Algorithm 4.

Algorithm 4 Collaborative Decryption (CD) (n servers)

```
Input: $S_1 : sk_1, S_2 : sk_2, \cdots, S_n : sk_n$ (private), $\mathsf{e}(g_2^{\alpha})$ where
   $\alpha \in \{0,1\}$, $q', g_2, \mathbb{G}, pk$ (public)
Output: $\alpha$.
1: let $(A_0, B_0) = \mathsf{E}(g^{\alpha})$
2: for $k = 1$ to $n-1$ {
3:    $S_k$ randomly chooses $r_k$ from $\mathbb{Z}_{\rho}^*$, computes
         $$A_k = A_{k-1}^{r_k}, B_k = (B_{k-1}/A_{k-1}^{sk_k})^{r_k},$$
      and sends $(A_k, B_k)$ to the server $S_{k+1}$.
4: }
5: The server $S_n$ computes $Z = B_{n-1}/A_{n-1}^{sk_n}$.
6: if $Z = 1$ { return 0 }
7: else { return 1 }
```

Theorem 4 (Collaborative Decryption Correctness). In Algorithm 4, the output is α.

Proof. When $\alpha = 0$, let $(A_0, B_0) = (g_2^{r_0}, pk^{r_0})$, we have

$$\begin{aligned} A_1 &= A_0^{r_1} = g_2^{r_0 r_1} \\ B_1 &= (B_0/A_0^{sk_1})^{r_1} = ((\prod_{l=1}^{n} pk_l)^{r_0}/g_2^{r_0 sk_1})^{r_1} = (\prod_{l=2}^{n} pk_l)^{r_0 r_1} \end{aligned}$$

By induction hypothesis, we assume that for $1 \leq k \leq n-1$,

$$\begin{aligned} A_k &= A_{k-1}^{r_k} = g_2^{r_0 r_1 \cdots r_k} \\ B_k &= (B_{k-1}/A_{k-1}^{sk_1})^{r_k} = (\prod_{l=k+1}^{n} pk_l)^{r_0 r_1 \cdots r_k} \end{aligned}$$

Therefore,

$$Z = B_{n-1}/A_{n-1}^{sk_n} = pk_n^{r_0 r_1 \cdots r_{n-1}}/g_2^{r_0 r_1 \cdots r_{n-1} sk_n} = 1,$$

and the output is 0. When $\alpha = 1$, we have $Z = g_2^{r_1 r_2 \cdots r_{n-1}} \neq 1$ and the output is 1. The theorem is proved. $\triangle$

If the output of Algorithm 4 is 0, the n servers cooperate to encrypt CI_i with the public key PK_U of the query user U by running Algorithm 5, where $\mathsf{E}(g_1^m, PK)$ and $\mathsf{E}(g_1^m, PK_U)$ stand for encryptions described in the initialisation of g_1^m under the public keys PK and PK_U, respectively.

Theorem 5 (Collaborative Encryption Correctness). In Algorithm 5, (A, B) is an ElGamal encryption of $g_1^{m_1+m_2+\cdots+m_n}$ with the public key PK_U.

Algorithm 5 Collaborative Encryption (CE) (n servers)

```
Input: $S_1 : SK_1, S_2 : SK_2, \cdots, S_n : SK_n$ (private), $\mathsf{E}(g_1^m, PK), N, g, PK_U$ (public)
Output: $\mathsf{E}(g_1^m, PK_U)$.
1: given $\mathsf{E}(g_1^m, PK)$, the $n$ servers cooperate to run Algorithm 1 to privately share $m$ and then each server $S_k$ privately holds $m_k$ such that $\sum_{k=1}^n m_k = m$.
2: for $k = 1$ to $n$ {
3:     $S_k$ randomly chooses $r_k$ from $\mathbb{Z}_\rho^*$, computes
           $A_k = g^{r_k}(mod\ N^2), B_k = g_1^{m_k} PK_U^{r_k}(mod\ N^2)$
       and sends $(A_k, B_k)$ to the social network service provider.
4: }
5: The service provider computes
           $A = \prod_{k=1}^n A_k(mod\ N^2), B = \prod_{k=1}^n B_k(mod\ N^2)$
6: return $(A, B)$
```

Proof. According to Algorithm 5, we have

$$A = \prod_{k=1}^n A_k = \prod_{k=1}^n g^{r_k} = g^{r_1+r_2+\cdots+r_n}$$

$$B = \prod_{k=1}^n B_k = \prod_{k=1}^n g_1^{m_k} PK_U^{r_k} = g_1^{m_1+\cdots+m_n} PK_U^{r_1+\cdots+r_n}$$

Therefore, the theorem is proved. △

After receiving $\mathsf{E}(g_1^{CI_i}, PK_U)$ from the service provider, the query user U runs Algorithm 6 to obtain CI_i.

Algorithm 6 Decryption (User U)

```
Input: $\mathsf{E}(g_1^y, PK_U), N, g, PK_U$ (public), $U : SK_U$ (private)
Output: $y$.
1: let $(A, B) = \mathsf{E}(g_1^y, PK_U)$
2: The query user $U$ computes
           $x = [B/A^{SK_U}(mod\ N^2) - 1]/N$
3: return $x$
```

Theorem 6 (Decryption Correctness). In Algorithm 6, we have $x = y(mod\ N)$.

Proof. Assume that $A = g^r$ and $B = g_1^y PK_U^r$, we have

$$\begin{aligned} x &= [B/A^{SK_U}(mod\ N^2) - 1]/N \\ &= [g_1^y PK_U^r/(g^r)^{SK_U}(mod\ N^2) - 1]/N \\ &= ((1+N)^y(mod\ N^2) - 1)/N \\ &= (1 + y(mod\ N)N - 1)/N = y(mod\ N) \end{aligned}$$

The theorem is proved. △

Because $CI_i < N$, the output of Algorithm 6 is CI_i if the input is $\mathsf{E}(g_1^{CI_i}, PK)$.

The above process is repeated until t matching users are found from DB and returned to the querying user. In case that the total number of matching users from DB is less than t, all matching users are returned to the querying user.

6 Extended Profile Matching Protocol

6.1 User Profile Matching for Multiple Thresholds

In the user profile matching protocol described in Section 5, the query user only specify single threshold δ only. In some user matching queries, the user may wish to specify different thresholds for different groups of attributes, respectively. For example, the query user may wish to find people with age between 20 and 30 and height between 170cm and 180cm.

In this section, we extend the user profile matching protocol for a single threshold to allow the query user to specify multiple thresholds. Let us consider two thresholds at first.

Assume that the query user U chooses two groups of attributes, one group consists of attributes $A_1, A_2, \cdots, A_m$ and another group contains attributes $A'_1, A'_2, \cdots, A'_{m'}$ and wishes to find a matching user U_i such that $d_{A_1,A_2,\cdots,A_m}(U_i, U) = \sum_{j=1}^m w_j(a_{ij} - a_j)^2 \le \delta$, $d_{A'_1,A'_2,\cdots,A'_{m'}}(U_i, U) = \sum_{j=1}^{m'} w'_j(a'_{ij} - a'_j)^2 \le \delta'$, where a_{ij} and a'_{ij} are the value of the attributes A_j and A'_j, respectively, for the user U_i, and $a_j, w_j, \delta, a'_j, w'_j, \delta'$ are specified by U in the query.

First of all, the user U encrypts $g_1^{a_1}, \cdots, g_1^{a_m}, g_1^{a'_1}, \cdots, g_1^{a'_m}, g_1^{w_1}, \cdots, g_1^{w_m}, g_1^{w'_1}, \cdots, g_1^{w'_m}, g_1^{\delta}, g_1^{\delta'}$ with the ElGamal encryption scheme and PK as described in Section 5.1. In addition, the user U randomly chooses the private key SK_U and computes the public key $PK_U = g^{SK_U}$. Then the user generates a query
$Q = \{A_1, \mathsf{E}(g_1^{a_1}), \mathsf{E}(g_1^{w_1}), \cdots, A_m, \mathsf{E}(g_1^{a_m}), \mathsf{E}(g_1^{w_m}), \mathsf{E}(g_1^{\delta}), A'_1, \mathsf{E}(g_1^{a'_1}), \mathsf{E}(g_1^{w'_1}), \cdots, A'_{m'}, \mathsf{E}(g_1^{a'_m}), \mathsf{E}(g_1^{w'_m}), \mathsf{E}(g_1^{\delta'}), PK_U\}$
and submits Q to the service provider.

Given Q, the n matching servers cooperate to run Algorithm 1 repeatedly to privately share $a_1, w_1, \cdots, a_m, w_m, a'_1, w'_1, \cdots, a'_{m'}, w'_{m'}$.

For each user U_i in the database, the n matching servers cooperate to determine whether U_i meets the thresholds specified by the query user U in three steps as follows.

Step 1: For attributes $A_1, A_2, \cdots, A_m$, the n matching servers collaborate to compute $\mathsf{E}(g_1^{\delta - d_{A_1,\cdots,A_m}(U,U_i)})$ as described in Section 5.2. Then, the n servers cooperate to privately share $\delta - d_{A_1,\cdots,A_m}(U, U_i)$ without revealing it by running Algorithm 1 and to determine if $\delta - d_{A_1,\cdots,A_m}(U, U_i) \ge 0$ by running Algorithm 3. Let the output of Algorithm 3 be $\mathsf{e}(g_2^{\alpha})$, where $\alpha = 0$ if $\delta - d_{A_1,\cdots,A_m}(U, U_i) \ge 0$ and 1 otherwise.

Step 2: For attributes $A'_1, A'_2, \cdots, A'_{m'}$, the n matching servers collaborate to compute $\mathsf{E}(g_1^{\delta' - d_{A'_1,\cdots,A'_{m'}}(U,U_i)})$ as described in Section 5.2. Then, the n servers cooperate to privately share $\delta' - d_{A'_1,\cdots,A'_{m'}}(U, U_i)$ without revealing it by running Algorithm 1 and to determine if $\delta' - d_{A'_1,\cdots,A'_{m'}}(U, U_i) \ge 0$ by running Algorithm 3. Let the output of Algorithm 3 be $\mathsf{e}(g_2^{\alpha'})$, where $\alpha' = 0$ if $\delta' - d_{A'_1,\cdots,A'_{m'}}(U, U_i) \ge 0$ and 1 otherwise.

Step 3: After obtaining $\mathsf{e}(g_2^{\alpha})$ and $\mathsf{e}(g_2^{\alpha'})$, the n servers cooperate to compute

$$\mathsf{e}(g_2^{\alpha\alpha'}) = \mathsf{e}(g_2^{((2\alpha-1)\alpha'+\alpha')2^{-1}}) = (\mathsf{e}(g_2^{(2\alpha-1)\alpha'})\mathsf{e}(g_2^{\alpha'}))^{2^{-1}}.$$

Note that $\mathsf{e}(g_2^{(2\alpha-1)\alpha'})$ can be computed with a conditional gate as described in Section 3.2. Then the n servers cooperate to compute

$$\mathsf{e}(g_2^{(1-\alpha)(1-\alpha')}) = \mathsf{e}(g_2)\mathsf{e}(g_2^{\alpha\alpha'})/(\mathsf{e}(g_2^{\alpha})\mathsf{e}(g_2^{\alpha'})),$$

and decrypt $\mathsf{e}(g_2^{(1-\alpha)(1-\alpha')})$ by running Algorithm 4. If the output is 1 (i.e., $\mathsf{e}(g_2^{(1-\alpha)(1-\alpha')}) = \mathsf{e}(g_2)$ and thus $\alpha = \alpha' = 0$), the n servers cooperate to encrypt CI_i by running Algorithm 5 and the social networking service provider returns $\mathsf{E}(g_1^{CI_i}, PK_U)$ as the response to the query user U. After receiving $\mathsf{E}(g_1^{CI_i}, PK_U)$ from the service provider, the query user U runs Algorithm 6 to obtain CI_i.

The above process is repeated until t matching users are found and returned to the querying user. In case that the total number of matching users is less than t, all matching users are returned to the querying user.

Now we consider multiple thresholds for user matching in general. The query user chooses several groups of attributes, specifies a dissimilarity threshold for each group of attributes and logical relations (including NOT, AND, OR) for the thresholds, generates a query Q accordingly and sends it to the social service provider.

Given the query Q and the profile of a user U_i, the n matching servers cooperate to perform Step 1 for each group of attributes and each threshold to obtain $\mathsf{e}(g_2^{\alpha})$, where $\alpha = 0$ if the dissimilarity threshold is satisfied and 1 otherwise. Based on the logical relations specified by the query user U and $\mathsf{e}(g_2^{\alpha}), \mathsf{e}(g_2^{\alpha'}), \cdots$ obtained in Step 1 (Step 2), the n matching servers cooperate to evaluate the logical conditions like Step 3 with conditional gates described in Section 3.2.

When the output of Algorithm 5 meets the logical condition, the n matching servers cooperate to compute $\mathsf{E}(g_1^{CI_i}, PK_U)$ and the social network provider returns it as the response to the querying user U and then the user U decrypts the response to obtain CI_i like Step 3.

6.2 User Profile Matching for Categorical Attributes

For numerical attributes, we define dissimilarities between two users as in Section 4. These dissimilarity definitions are not applicable to categorical attributes, e.g., user nationality, user hobbies, and etc. However, for some special queries, e.g., finding users with the same nationality, we can re-use our user profile matching protocol as follows.

Assume that $A_1, A_2, \cdots, A_m$ are categorical attributes. In the initialization, each user U_i maps the value of each categorical attribute A_j to the data a_{ij} of a fixed size by a hash function H, encrypts the $a_{i1}, a_{i1}^2, \cdots, a_{im}, a_{im}^2$ as described in Section 5.1, and then sends the ciphertexts $\mathsf{E}(g_1^{CI_i}), \mathsf{E}(g_1^{a_{i1}}), \mathsf{E}(g_1^{a_{i1}^2}), \cdots, \mathsf{E}(g_1^{a_{im}}), \mathsf{E}(g_1^{a_{im}^2})$ to the social networking service provider, which stores them in the user profile database DB.

When a user wishes to find people such that $a_{ij} = a_j$ ($j = 1, 2, \cdots, m$), he sends to the service provider a query

$$Q = \{\mathsf{E}(g_1^{a_1}), w_1 = 2, \cdots, \mathsf{E}(g_1^{a_m}), w_m = 2, \delta = 1, PK_U\}.$$

Like our protocol described in Section 5.2, the n matching servers can cooperate to compute $\mathsf{E}(g_1^{(a_j - a_{ij})^2})$ for $j = 1, 2, \cdots, m$ on the basis of Q and the profile of a user U_i in the database. Then the n servers can compute $\prod_{j=1}^{m} \mathsf{E}(g_1^{(a_j-a_{ij})^2})^{w_j} = \mathsf{E}(g_1^{\sum_{j=1}^{m} 2(a_j-a_{ij})^2})$ and $\mathsf{E}(g_1^{1-\sum_{j=1}^{m} 2(a_j-a_{ij})^2})$.

Next, the n servers cooperate to share $d = 1 - \sum_{j=1}^{m} 2(a_j - a_{ij})^2$ by running Algorithm 1 and then determine if $d \geq 0$ by running Algorithm 3. Assume that the output of Algorithm 3 is $\mathsf{e}(g_2^{\alpha})$. If $\alpha = 0$, then $d = 1 - \sum_{j=1}^{m} 2(a_j - a_{ij})^2 \geq 0$ and therefore $a_{ij} = a_j$ for $j = 1, 2, \cdots, m$.

The output of Algorithm 3, i.e., $\mathsf{e}(g_2^{\alpha})$, may be incorporated with other dissimilarity thresholds for numerical attributes as described in Section 6.1 to determine if the profile of the user U_i meets the searching criteria.

7 Security Analysis

7.1 Security of Our Underlying Encryption Scheme

To compute the distance privately, we use the ElGamal encryption scheme [8], but we choose the public parameters on the basis of the Paillier encryption scheme [21]. The public parameters are (N, g), where N is a product of two larger primes q and p and $g = (1+N)^p r^N (mod\ N^2)$ for a randomly chosen integer $r \in \{2, 3, \cdots, N-1\}$. It is not hard to see $g^{(p-1)(q-1)} \neq 1(mod\ N^2)$, but $g^{q(p-1)(q-1)} = 1(mod\ N^2)$. Thus the order g is more than q. In addition, because $(g-1)/N$ is not an integer, there is no integer a such that $g = (1+N)^a (mod\ N^2)$.

The security of the original ElGamal encryption scheme is built on the decisional Diffie-Hellman (DDH) problem: given g^a, g^b, g^c, determine if $c = ab$. If the DDH problem is hard, the ElGamal encryption scheme is semantically secure.

Because $g = (1+N)^p r^N$, we have $g^x = (1+N)^{px} r^{Nx}$, which is a Paillier encryption of $px(mod\ N)$. If p and q are given, one can decrypt g^x to obtain px. So given g, g^a, g^b, g^c in a DDH problem, one can obtain $p, pa(mod\ N), pb(mod\ N), pc(mod\ N)$, by which it is easy to see if $c = ab$. Because p and q are assumed to be public in [33], the variant ElGamal encryption scheme has been shown to be insecure [22].

In this paper, we assume that (N, g) can be generated so that no one knows p and q, e.g., by the approach in [14] or by the Intel Software Guard Extensions (Intel SGX). Furthermore, there is no integer a such that $g = (1+N)^a (mod\ N^2)$ and it is well-known that the factorization of a large integer N is a hard problem. Therefore, under the assumption that no one knows p and q, the DDH problem over the group $\mathbb{G} = \{g^{\alpha}(mod\ N^2) | \alpha \in \mathbb{Z}_N\}$ is hard.

In the computation of the distance, our encryption scheme is actually a combination of the ElGamal and Paillier encryption schemes. Like the Paillier scheme, from an encryption ($A = g^r(mod\ N^2), B = g_1^x PK^r(mod\ N^2)$) of x, one can obtain $B^N = (g_1^x)^N PK^{rN} = PK^{rN}(mod\ N^2)$. However, given X^N, one cannot distinguish X with a random integer $R \in \mathbb{Z}_{N^2}^*$. Otherwise, the Paillier scheme is not semantically secure.

In summary, under the assumption that no one knows p and q and the Paillier scheme is semantically secure, the ElGamal scheme described in the initialization is semantically secure.

7.2 User Profile Privacy

In Section 4, we define user profile privacy with a game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. We assume that the adversary $\mathcal{A}$ has compromised $n-1$ matching servers $S_2, S_3, \cdots, S_n$ and known their private keys SK_k for $k = 2, 3, \cdots, n$.

The adversary $\mathcal{A}$ chooses two user profiles $(a_{01}, a_{02}, \cdots, a_{0m})$ and $(a_{11}, a_{12}, \cdots, a_{1m})$ and sends them to the challenger $\mathcal{C}$. The challenger $\mathcal{C}$ chooses a random bit $b \in \{0, 1\}$, and executes the Profile Generation (PG) to obtain an encrypted profile $P_b = \mathsf{PG}(PK, a_{b1}, a_{b2}, \cdots, a_{bm})$, and then sends P_b back to the adversary $\mathcal{A}$.

According to the initialization described in Section 5.1,

$$P_b = \{\mathsf{E}(g_1^{a_{b1}}), \mathsf{E}(g_1^{a_{b1}^2}), \cdots, \mathsf{E}(g_1^{a_{bm}}), \mathsf{E}(g_1^{a_{bm}^2})\}.$$

For simplicity, we do not include the contact information here. It can be considered as an attribute.

Given $\mathsf{E}(g_1^x) = (g^r(mod\ N^2), g_1^x PK^r(mod\ N^2))$, the adversary can compute $g_1^x PK^r/(g^r)^{SK_2+\cdots+SK_n} = g_1^x PK_1^r$. Without knowing the private key SK_1 of the first server, the adversary cannot distinguish x with any other value because the ElGamal encryption scheme is semantically secure. Therefore, the adversary advantage in guessing b is negligible. According to Definition 4, our protocols (including the extended protocols) have user profile privacy.

7.3 User Query Privacy

In Section 4, we also define user query privacy with a game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. We also assume that the adversary $\mathcal{A}$ has compromised $n-1$ matching servers $S_2, S_3, \cdots, S_n$ and known their private keys SK_k, sk_k for $k = 2, 3, \cdots, n$. In addition, we assume that the adversary is different from the adversary in the user profile privacy game.

The adversary $\mathcal{A}$ chooses two user profiles $(a_{01}, a_{02}, \cdots, a_{0m})$ with corresponding weights $(w_{01}, w_{02}, \cdots, w_{0m})$ and dissimilarity threshold δ_0, and $(a_{11}, a_{12}, \cdots, a_{1m})$ with corresponding weights $(w_{11}, w_{12}, \cdots, w_{1m})$ and dissimilarity threshold δ_1, and sends them to the challenger $\mathcal{C}$. The challenger $\mathcal{C}$ chooses a random bit $b \in \{0, 1\}$, and executes the Query Generation (QG) to obtain a query $Q_b = \mathsf{QG}(PK, a_{b1}, w_{b1}, a_{b2}, w_{b2}, \cdots, a_{bm}, w_{bm}, \delta_b)$, and then sends Q_b back to the adversary $\mathcal{A}$.

According to our protocol,

$$Q_b = \{\mathsf{E}(g_1^{a_{b1}}), \mathsf{E}(g_1^{w_{b1}}), \cdots, \mathsf{E}(g_1^{a_{bm}}), \mathsf{E}(g_1^{w_{bm}}), \mathsf{E}(g_1^{\delta_b}), PK_U\}$$

where $PK_U = g^{SK_U}$ is randomly generated.

As the analysis on user profile privacy, given $\mathsf{E}(g_1^x)$, the adversary cannot distinguish x with any other value without knowing SK_1.

In the private sharing algorithm, given $\mathsf{E}(g_1^x)$ and $(g_1^{x_1} A^{SK_1})^{-1}$ from the first server, the adversary can obtain $g_1^{x-x_1}$ and then $x - x_1$. However, x_1 is sufficiently large, i.e., around $2^{\ell_x+\ell}$, to hide all possible values of x, where ℓ_x is the bit-length of x and ℓ is a sufficiently large statistical security parameter. Therefore, the adversary still cannot distinguish x from any other value. In addition, the adversary cannot retrieve A^{SK_1} from $(g_1^{x_1} A^{SK_1})^{-1}$ in view of the existence of x_1. Thus the adversary cannot use A^{SK_1} to decrypt any other ciphertext (A, B) in case that the adversary decoys the first server to output A^{SK_1}.

In the private multiplication algorithm (Algorithm 2), there is no decryption of any query component and thus it does not help the adversary win the game.

In the private comparison algorithm (Algorithm 3), the encryption scheme used to encrypt 0 or 1 only is different from the encryption scheme used to encrypt user profiles and queries. There is no decryption of any query component in the algorithm and it does not help the adversary win the game.

In the collaborative decryption algorithm (Algorithm 4), there is only one decryption. The algorithm is designed so that the decryption result is either 0 or 1 and it is hard for the adversary to retrieve $A_0^{sk_1}$ from $(B_0/A_0^{sk_1})^{r_k}$, where r_k is randomly chosen by the first server. Thus this algorithm also does not help the adversary win the game.

In the collaborative encryption algorithm (Algorithm 5), the private sharing algorithm (Algorithm 1) is used to share the contact information CI_i of the user U_i at first. As we have shown, Algorithm 1 does not help the adversary win the game. After that, the first server encrypts $CI_i^{(1)}$ with the public key PK_U of the query user U. Without knowing the private key SK_U, the adversary cannot get $CI_i^{(1)}$ and hence is not able to determine CI_i. Thus the adversary cannot distinguish CI_i in $\mathsf{E}(g_1^{CI_i})$ from any other value and cannot make use of CI_i to guess b.

In summary, our user profile matching protocol is composed of Algorithms 1-6. Algorithm 6 is executed by the query user. Algorithms 1-5 are cooperatively executed by the n matching servers. Even if the adversary compromises $n-1$ servers, he still cannot distinguish the two queries in the game. Therefore, the adversary's advantage in guessing b is negligible. According to Definition 5, our protocols (including the extended protocols) have user query privacy.

8 Performance Analysis

8.1 Theoretic Performance Analysis

We need two instances of ElGamal encryption for our protocols. The first instance, described in Section 5.1, is used in Algorithms 1, 2, 5, and 6, since in these four algorithms the value of the exponent of the generator in the cyclic group has to be retrieved by a decrypting party. The second instance is the original ElGamal encryption scheme, used in Algorithms 3 and 4 to encrypt a bit.

We construct the first instance of ElGamal encryption by creating an RSA modulus N, which is the product of two large primes p and q with bit-length equal to 512-bit each. The generator g is chosen to be $(1+N)^p r^N$ modulo N^2, which has an order more than q under $\mathbb{Z}_{N^2}^*$. As for the second instance, we use the original ElGamal encryption scheme [8] based on an order q' subgroup modulo p', where p' and q' are prime numbers of 1,024-bit and 160-bit, respectively. The generator

TABLE 1: Computation Complexities per Server

Protocols	Distance Computation	Private Comparison	Response Generation
Basic Protocol [32]	$11m$ (Exp.)	1 (Exp.)	3 (Exp.)
Privacy-Enhanced Protocol [32]	$18m$ (Exp.)	$O(\lambda)$ (exp.)	3 (Exp.)
New Proposed Protocol	$8m$ (Exp.)	$O(\lambda)$ (exp.)	5 (Exp.)

g_2 for this instance is chosen to be an element in $\mathbb{Z}^*_{p'}$ with order equal to q'. The first instance allows a decrypting party to compute the discrete logarithm to the base of g_1 modulo N^2 but this comes at a cost of a higher decryption complexity since N^2 is a 2,048-bit integer.

The most important functionality we provide in our protocol is query processing, which in turn consists of three sub-protocols. An encrypted weighted and squared Euclidean distance between the given record (query) and a profile is jointly computed by those n matching servers in the first sub-protocol, whose output is then fed to the second sub-protocol to determine whether or not there is a match. If both match, a response, i.e., the encrypted contact information of the matching user, is generated and returned to the query user in the third sub-protocol. Therefore, in the following performance analysis, we focus on the performance analysis of the efficiency of our proposed protocols in distance computation, private comparison and response generation. We list the computation complexities of each server in the three sub-protocols for the basic protocol [32], the privacy-enhanced protocol [32] and the proposed new protocol in Table 1, where the user profile contains m attributes, and Exp. and exp. denote the modular exponentiation in the first instance and the second instance of the ElGamal encryption scheme, respectively, and λ denotes the max bit-length of the distance shares.

Note that both the basic and privacy-enhanced protocols [30] require each matching server to share the user profiles in the initialization and maintain a user database, while the new protocol keeps the user profiles in the service provider only. The basic protocol [32] does not hide the contact information of users, the weights of the query, and the difference between the distance and threshold. The privacy-enhanced protocol [32] hides them, but its private sharing algorithm requires each server to encrypt the share, broadcast the encrypted share, and verify the commitments from other servers and then collaborate to decrypt a value. In the new protocol, the private sharing algorithm only needs each server to send one decryption part to the last server.

8.2 Experimental Performance Analysis

Now, we have conducted extensive experiments on a real dataset[2] to evaluate the performance of our proposed protocols under different parameter settings. We removed the records with missing values and randomly sample 5,000 records from this dataset. We then choose *age*, *hours-per-week*, *capital-gain*, *capital-loss*, and *fnlwgt* as the attributes involved in the profile matching. Table 2 gives the original domain ranges of these attributes. As we can see from Table 2, the ranges for the attributes differ a lot. In order to ease profile matching, the profile data owners could first scale each of their attribute values to a predefined range, e.g., $[0, 2^{\ell_0} - 1]$ that is publicly known. We adopt this approach in our experiments.

TABLE 2: Domain Ranges of Attributes

Attribute	Type	Lower Bound	Upper Bound
age	Numerical	0	100
hours-per-week	Numerical	0	100
capital-gain	Numerical	0	100,000
capital-loss	Numerical	0	5,000
fnlwgt	Numerical	0	1,500,000

TABLE 3: Parameters in the Experiments

Parameter	Meaning	Values
$\lvert\mathcal{D}\rvert$	Dataset Size	1k **3k** 5k
ℓ_0	Max Bit-Length of Attribute Values	**5**
m	Number of Attributes	1 **3** 5
$\lvert w\rvert$	Max Bit-Length of Weight Values	**3**
ℓ	Statistical Security Parameter	20 **30** 40
n	Number of Matching Servers	**2** 4 6

In our experiments, we use several parameters summarized in Table 3. The default parameters are shown in boldface. In the following subsections, we focus on the empirical analysis of the efficiency of our proposed protocols.

The prototype of our system is implemented in C/C++, and the experiments were carried out on a computer equipped with an Intel i7-4770HQ CPU with 16 GB RAM running on Ubuntu 64-bit 14.04.1 operating system. We use the GNU MP library[3] to implement the ElGamal encryption scheme instances.

1) Distance Computation

In the first part of query processing, to enable n matching servers to privately determine whether or not there is a match in the next sub-protocol, a ciphertext $\mathsf{E}(g_1^{\delta - d(U,U_i)})$ should be computed and shared among the n matching servers, where $d(U, U_i) = \sum_{j=1}^{m} w_j(a_j - a_{ij})^2$.

First of all, we note that ℓ and n are the most decisive parameters regarding the security of our proposed solutions and hence we would like to study their effects on the efficiency of the protocols. In Figure 2(a) and Figure 2(b), we provide the evaluation time needed for secure distance computation when varying (ℓ, m) and $(n, |\mathcal{D}|)$, respectively. We include the results from the Basic protocol, the Privacy-Enhanced protocol and the New Proposed protocol in these figures and use *B*, *PE* and *NP* to denote them, respectively.

From both Figure 2(a) and Figure 2(b), it is clear that the time for distance computation grows linearly in both the number of attributes (m) and the number of records ($|\mathcal{D}|$) in the database, which is as expected because the computation of the Euclidean distance is performed in a componentwise manner and the computation of the Euclidean distance has to be done for each data record given a query.

On the other hand, it can be seen that the protocol NP is the most efficient protocol among those three being compared under the same configuration of (ℓ, m) or $(n, |\mathcal{D}|)$. There are several reasons. First, recall that in [32], the secret key SK_k of the server S_k used for the private sharing and private multiplication

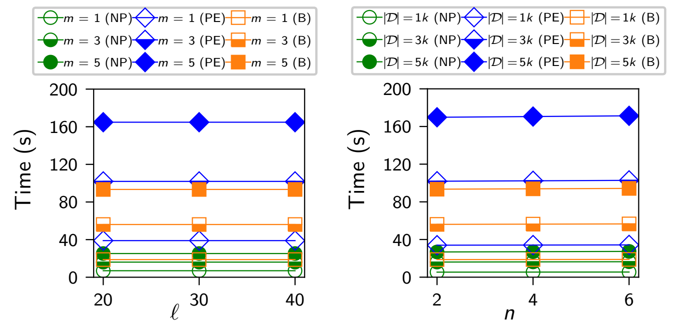
2. http://archive.ics.uci.edu/ml/datasets/Census+Income

3. https://gmplib.org/

was chosen from a much larger domain of $\{1, \cdots, q-1\}$, where q is a 512-bit prime number, while in the protocol NP, each matching server S_k randomly selects a secret key SK_k from $\{1, \cdots, \rho-1\}$, where $\rho = 2^{160}$. Second, the efficiency of the private sharing algorithm in the protocol NP is improved. Specifically, in the private sharing algorithm of the PE protocol, there are 2 heavy operations, i.e., one ElGamal encryption for $\mathsf{E}(g_1^{x_k})$ and one exponentiation for $C_1^{SK_k}$. The former requires the computation of g^r and PK^r for a 512-bit random integer r. In total, at least 3 exponentiations modulo N^2 are needed. However, in the new private sharing algorithm, each server only needs to compute $g_1^{x_k} A^{SK_k} (\bmod\ N^2)$ and send its inverse to the last server. Third, fewer invocations of the private multiplication algorithm are required in the protocol NP. To be precise, for the protocol B and PE, $\mathsf{E}(g_1^{a_j^2})$ and $\mathsf{E}(g_1^{a_j a_{ij}})$ are derived separately via two invocations of private multiplications, whereas in the new protocol, only one private multiplication on input $\mathsf{E}(g_1^{(a_j-2a_{ij})})$ and shares of a_j is required to compute $\mathsf{E}(g_1^{(a_j-2a_{ij})a_j})$. The protocol B is more efficient than the PE protocol in that w_j is publicly known in the former so that no further invocations involving expensive exponentiations are needed to produce $\mathsf{E}(g_1^{\delta-d(U,U_i)})$, whereas one additional private sharing to share w_j and a subsequent call to private multiplication for producing $\mathsf{E}(g_1^{w_j(a_j-a_{ij})^2})$ are necessary for the latter. Lastly, unlike the protocol B or PE, our new protocol reduces the overhead of re-encryption in the private multiplication protocol by each server S_k performing only one re-encryption on its own share of $\sum_{j=1}^m w_j^{(k)}(a_j-a_{ij})^2$, saving up to $2(m-1)$ exponentiations modulo N^2 per query, where $w_j^{(k)}$ denotes the share of w_j possessed by the server S_k. Overall, the protocol NP saves at least 82% of the computational overhead than the PE protocol.

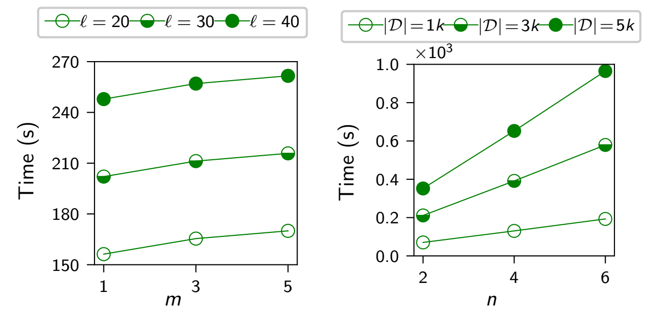
Furthermore, in Figure 2(a) and Figure 2(b), we observe that varying ℓ and n has very limited effects on the elapsed time for each of the protocols being compared. This is as expected. First, recall that when the statistical security parameter ℓ is increasing, a random share x_k possessed by the server S_k has to be enlarged as well to ensure that the share retrieved by the last server S_n would look like a random integer chosen from a much larger domain than the domain in which x, the integer to be shared, resides. In this regard, when the private multiplication algorithm is invoked later, the computational overhead would also become higher in that each component of $\mathsf{E}(g_1^y)$ has to be raised to a larger exponent x_k. However, due to the fact that the range of the bit-length of each share x_k in our experiment is much smaller (from 35 to 58 given the current possible parameter configurations) than the random integer r used in encrypting $g_1^{x_k}$, increasing ℓ only slightly increases the time for computing $\mathsf{E}(g_1^y)^{x_k}$.

In addition, we note that in both the private sharing and the private multiplication algorithms used in each protocol under consideration, the server S_k does not have to wait for other matching servers to compute $g_1^{x_k} A^{SK_k} \pmod{N^2}$ or a re-encryption of $\mathsf{E}(g_1^{x_k y})$, respectively. Thus, these expensive operations are highly parallelizable and the elapsed time does not depend on the number of matching servers so much.



(a) Varying ℓ and m ($|\mathcal{D}| = 3k$, $\ell_0 = 5$, $|w| = 3$, $n = 2$) (b) Varying n and $|\mathcal{D}|$ ($\ell_0 = 5$, $m = 3$, $|w| = 3$, $\ell = 30$)

Fig. 2: Evaluation Time for Distance Computation



(a) Varying m and ℓ ($|\mathcal{D}| = 3k$, $\ell_0 = 5$, $|w| = 3$, $n = 2$) (b) Varying n and $|\mathcal{D}|$ ($\ell_0 = 5$, $m = 3$, $|w| = 3$, $\ell = 30$)

Fig. 3: Evaluation Time for Private Comparison

2) Private Comparison

In this subsection, we report the time for determining whether there is a match between the given query and a profile. In the protocol B, it is allowed that the difference between $d(U, U_i)$ and δ is revealed. Hence, only one distributive decryption of $\mathsf{E}(g_1^{\delta-d(U,U_i)})$ is needed, which is much cheaper than the cost in the case when this difference has to be hidden, i.e., the protocols PE and NP. We thus study the effects of varying (m, ℓ) and $(n, |\mathcal{D}|)$ on the evaluation time when only the single bit indicating the matching result is allowed to be revealed. The results are reported in Figures 3(a) and 3(b), respectively. Notice that for the private comparison, both protocols PE and NP use the same private adder, which in turn is based on the same construction of conditional gate, and therefore we do not distinguish between them in those two figures.

As what we have seen in the distance computation, in Figure 3(b), it is clear that the evaluation time grows linearly in $|\mathcal{D}|$, the database size. However, unlike what is shown before, when ℓ, m, and n increase, we have a noticeable increase in evaluation time and such an increase is much higher for larger values of ℓ and n. The reason is the use of the cryptographic primitive for implementing the secure comparison, namely, the multiplication protocol based on the conditional gate.

To see this, recall that the weighted squared distance is $\sum_{j=1}^m w_j(a_j - a_{ij})^2$, which is upper-bounded by

$\sum_{j=1}^{m} w_j(2^{\ell_0}-1)^2$. Let ℓ' be the minimum number of bits required to represent this maximum distance, it can be seen that $\ell' = \lfloor \log_2((2^{\ell_0}-1)^2(\sum_{j=1}^{m} w_j)) + 1 \rfloor$. Hence, the private sharing (Algorithm 1) is invoked with the parameter $L = 2^{\ell'+\ell}$ to share the distance, which is upper-bounded by $2^{\ell'}-1$. Using a similar argument, it can be concluded that the minimum number of bits to represent a distance share owned by the server S_n is $\ell'' = \lfloor \log_2((2^{\ell'}-1)+(n-1)(2^{\ell'+\ell}-1)) + 1 \rfloor$. The minimum number of bits needed to represent S_n's distance share thus grows linearly in $(\ell_0+\ell)$. According to the fact that the time complexity of private addition increases linearly in the bit-length of its two input numbers, the evaluation time for private comparison grows linearly in ℓ, which is consistent with our experimental results in Figure 3(a). Increasing the number of attributes (m), however, does not have a linear effect on the execution time. The evaluation time only grows logarithmically in m, which can be confirmed with the formula representing ℓ' above.

In the evaluation of the conditional gate, we assume that the server S_i has to wait for its subsequent servers to finish the computation of U_n and V_n to proceed to the next stage in the conditional gate, implying that the evaluation time increases at least linearly in the number of matching servers involved in the conditional gate, which in turn is used to implement the private addition. This explains why we see a linear increase in n from Figure 3(b). However, if we allow those n servers to process n user profiles in parallel by interleaving the order of query execution for different user profiles, the evaluation time will reduce significantly, because the server S_i can compute U_i' and V_i' for another user profile in the database while waiting on U_n and V_n to be generated by its following servers.

8.3 Comparison with Shahandashti et al.'s Protocol

At last, we compare the performance of our protocol with Shahandashti et al.'s [24], where Alice and Bob, each holding a private fingerprint, find out if their fingerprints belong to the same individual.

In Shahandashti et al.'s protocol, let Alice have as private input a set of minutiae $F = \{p_1, p_2, \cdots, p_n\}$ where for $i = 1$ to n, $p_i = (t_i, x_i, y_i, \theta_i)$ and Bob have as private input a set of minutiae $F' = \{p_1', p_2', \cdots, p_m'\}$ where for $j = 1$ to m, $p_j' = (t_j', x_j', y_j', \theta_j')$. Alice's private output of the protocol is the binary predicate indicating whether or not there are at least τ of her minutiae matching those of Bob's, where two minutiae are defined to match if their types t_i and t_j' are the same, their locations are closer than a threshold Euclidean distance d_E, i.e., $\sqrt{(x_i - x_j')^2 + (y_i - y_j')^2} \le d_E$, and their orientations are within an angular difference d_a, i.e., $min(|\theta_i - \theta_j'|), 2\pi - |\theta_i - \theta_j'|) \le d_a$.

For comparison, in our protocol for multiple thresholds, we assume that there are two matching servers, which are given the encryptions of F and F' and output if two minutiae match or not. Let us denote the set of all possible minutia types by T, the set of all possible Euclidean distances (resp. angular differences) between two arbitrary points (resp. orientations) by D_E (resp. D_a). The computation and communication complexities of our two-party protocol and Shahandashti et al.'s are roughly compared in Table 4, where ℓ is the statistical security parameter in our protocol, κ is the number of matching, γ is the ciphertext size, and $|S|$ denotes the number of elements in the set S.

TABLE 4: Performance Comparison with Shahandashti et al.'s

Protocols	Computation Complexity	Communication Complexity
Our two-party Protocol	$3O(\kappa\ell)$ (exp.)	$3O(\kappa\ell)\gamma$
Shahandashti et al.'s Protocol	$(O(\kappa\|T\|)+O(\kappa\|D_E\|)$ $O(\kappa\|D_a\|))$ (exp.)	$(O(\kappa\|T\|)+O(\kappa\|D_E\|)$ $O(\kappa\|D_a\|))\gamma$

Usually, $\ell = 30$ is sufficient in our protocol. If $|T| = 10, |D_E| = 1{,}000, |D_a| = 10$ and $\kappa = 1$, Shahandashti et al.'s protocol needs to compute about 1,020 encryptions and exchange about 1,020 ciphertexts. However, our protocol needs to compute about 90 encryptions and exchange about 90 ciphertexts. Shahandashti et al.'s protocol is more efficient than our protocol only when $|T| + |D_E| + |D_a|$ is less than 90, in other words, the number of possible fingerprints is less than 27,000.

9 Conclusion

In this paper, we proposed a new solution for privacy- preserving user profile matching with homomorphic encryption technique and multiple servers. Our solution allows a user to find out the matching users with the help of multiple servers without revealing the query and the user profiles. Security analyses have showed that the new protocol achieves user profile privacy and user query privacy. The experimental results have showed that the new protocol is practical and feasible. Our future work is to improve the performance of computing conditional gates by parallel computation.

10 Acknowledgments

This work was supported by Australian Research Council Discovery Project (DP190102835), the National Research Foundation, Prime Minister's Office, Singapore under its Strategic Capability Research Centres Funding Initiative, and Data61 Research Collaboration Project.

References

[1] R. Agrawal, A. Evfimievski, and R. Srikant, Information sharing across private databases, in SIGMOD 2003, pp. 86-97.

[2] M. von Arb, M. Bader, M. Kuhn, and R. Wattenhofer, Veneta: Serverless friend-of-friend detection in mobile social networking, in IEEE WIMOB 2008, pp. 184-189.

[3] B. H. Bloom, Space/time trade-offs in hash coding with allowable errors, Communications of the ACM 13 (7): 422-426, 1970.

[4] D. Boneh, E. J. Goh, K. Nissim, Evaluating 2-DNF formulas on ciphertexts, in TCC 2006, pp 325-341.

[5] D. Chaum, Blind signatures for untraceable payments, in Crypto 1982, pp. 199-203.

[6] E. D. Cristofaro and G. Tsudik, Practical private set intersection protocols with linear complexity, in Financial Cryptography and Data Security 2010.

[7] D. Dachman-Soled, T. Malkin, M. Raykova, and M. Yung, Efficient robust private set intersection, in ACNS 2009, pp. 125-142.

[8] T. ElGamal, A public-key cryptosystem and a signature scheme based on discrete logarithms, IEEE Transactions on Information Theory 31 (4): 469-472, 1985

[9] M. Freedman, K. Nissim, and B. Pinkas, Efficient private matching and set intersection, in EUROCRYPT 2004, pp. 1-19.

[10] C. Gentry, Fully homomorphic encryption using ideal lattices, in STOC 2009, pp 169-178.

[11] S. Goldwasser and S. Micali, Probabilistic encryption and how to play mental poker keeping secret all partial information, in Proc. 14th Symposium on Theory of Computing, 1982, pp. 365-377.

[12] D. Harris, D. M. Harris and S. L. Harris, Digital Design and Computer Architecture, Morgan Kaufmann Publishers, 2007.

[13] C. Hazay and Y. Lindell, Efficient protocols for set intersection and pattern matching with security against malicious and covert adversaries, in TCC 2008, pp. 155-175.

[14] C. Hazay, G. L. Mikkelsen, T. Rabin, T. Toft, A. A. Nicolosi, Efficient RSA key generation and threshold paillier in the two-party setting, in CT-RSA 2012, pp. 313-331.

[15] Y. Huang, L. Malka, D. Evans and J. Katz, Efficient privacy-preserving biometric identification, in NDSS 2011.

[16] S. Jarecki and X. Liu, Efficient oblivious pseudorandom function with applications to adaptive OT and secure computation of set intersection, in TCC'09, 2009, pp. 577-594.

[17] R. Li and C. Wu, An unconditionally secure protocol for multi-party set intersection, in ACNS 2007, pp. 226-236.

[18] M. Li, N. Cao, S. Yu, and W. Lou, FindU: Privacy-preserving personal profile matching in mobile social networks, in IEEE INFOCOM 2010, pp. 1-9.

[19] L. Kissner and D. Song, Privacy-preserving set operations, in CRYPTO 2005, pp. 241-257.

[20] G. S. Narayanan, T. Aishwarya, A. Agrawal, A. Patra, A. Choudhary, and C. P. Rangan, Multi party distributed private matching, set disjointness and cardinality of set intersection with information theoretic security, in CANS 2009, pp. 21-40.

[21] P. Paillier, Public-key cryptosystems based on composite degree residuosity classes, in Proc. 17th Int. Conf. Theory Appl. Cryptograph. Techn., 1999, pp. 223-238.

[22] F. Y. Rao, On the security of a variant of ElGamal encryption scheme, IEEE Trans. Dependable and Secure Computing, 2017.

[23] Y. Sang, H. Shen, and N. Xiong, Efficient protocols for privacy preserving matching against distributed datasets, in ICICS 2006, pp. 210-227.

[24] S. F. Shahandashti, R. Safavi-Naini, P. Ogunbona, Private fingerprint matching, in ACISP 2012, pp 426-433.

[25] A. Shamir, How to share a secret, Communications of the ACM 22 (11): 612-613, 1979.

[26] B. Schoenmakers and P. Tuyls, Practical two-party computation based on the conditional gate, in ASIACRYPT 2004, pp. 119-136.

[27] L. Sun, L. Zhang, X. Ye, Randomized bit vector: Privacy-preserving encoding mechanism, in CIKM 2018, pp. 1263-1272.

[28] J. Vaidya and C. Clifton, Secure set intersection cardinality with application to association rule mining, J. Comput. Secur. 13(4): 593-622, 2005.

[29] Z. Yang, B. Zhang, J. Dai, A. Champion, D. Xuan, and D. Li, E-smalltalker: A distributed mobile system for social networking in physical proximity, in IEEE ICDCS, 2010.

[30] A. C. Yao, Protocols for secure computations, in SFCS 1982.

[31] Q. Ye, H. Wang, and J. Pieprzyk, Distributed private matching and set operations, in ISPEC 2008, pp. 347-360.

[32] X. Yi, E. Bertino, F. Y. Rao, A. Bouguettaya, Practical privacy-preserving user profile matching in social networks, in ICDE 2016, pp. 373-384.

[33] X. Yi, A. Bouguettaya, D. Georgakopoulos, A. Song. Privacy protection for wireless medical sensor data, IEEE Transactions on Dependable and Secure Computing, 13(3): 369-380, 2015.

PLACE PHOTO HERE

Xun Yi is a Professor with the Computer Science and Software Engineering, School of Science, RMIT University, Australia. His research interests include applied cryptography, computer and network security, mobile and wireless communication security, and data privacy protection. He has published more than 200 research papers in international journals and conference proceedings. He has ever undertaken program committee members for more than 30 international conferences.

PLACE PHOTO HERE

Elisa Bertino is a professor of computer science at Purdue University. Her main research interests include security, privacy, digital identity management systems, database systems, distributed systems, and multimedia systems. She is currently serving as EIC of IEEE Transactions on Dependable and Secure Computing. She received the 2002 IEEE Computer Society Technical Achievement Award for outstanding contributions to database systems and security.

PLACE PHOTO HERE

Fang-Yu Rao is a Ph.D. candidate in Computer Science at Purdue University. He received his Master and Bachelor of Computer Science and Engineering from National Sun Yat-sen University. His research interests are in data privacy, information security, and applied cryptography, with an emphasis on privacy-preserving data analytics.

PLACE PHOTO HERE

Kwok Yan Lam is a Professor with the School of Computer Science and Engineering, Nanyang Technological University, Singapore. He received his B.Sc. (First Class Honours) from the University of London in 1987 and his Ph.D. from the University of Cambridge in 1990. In 1997, he founded PrivyLink International Ltd, a spin-off company of the National University of Singapore, specializing in e-security technologies for homeland security and financial systems.

PLACE PHOTO HERE

Surya Nepal is a Principal Research Scientist at CSIRO Data61, Australia. His main research interest include the development and implementation of technologies in the area of distributed systems and social networks, with a specific focus on security, privacy, and trust. At CSIRO, he undertook research in the area of social networks, security, privacy and trust in collaborative environment and cloud systems and big data.

PLACE PHOTO HERE

Athman Bouguettaya is a Professor and Head of School of Information Technology at The University of Sydney, Australia. He received his PhD in Computer Science from the University of Colorado at Boulder (USA) in 1992. He is a founding member and past President of the Service Science Society. He has published more than 200 books, book chapters, and articles in journals and conferences in the area of databases and service computing.