Fine-grained Private Matching for Proximity-based Mobile Social Networking

Rui Zhang*, Yanchao Zhang*, Jinyuan (Stella) Sun†, and Guanhua Yan‡
*Arizona State University, Tempe, AZ, USA
†University of Tennessee, Knoxville, TN, USA
‡Los Alamos National Laboratory, Los Alamos, NM, USA
*{ruizhang,yczhang}@asu.edu, †jysun@eecs.utk.edu, ‡ghyan@lanl.gov

***Abstract*—Proximity-based mobile social networking (PMSN) refers to the social interaction among physically proximate mobile users directly through the Bluetooth/WiFi interfaces on their smartphones or other mobile devices. It becomes increasingly popular due to the recently explosive growth of smartphone users. Profile matching means two users comparing their personal profiles and is often the first step towards effective PMSN. It, however, conflicts with users' growing privacy concerns about disclosing their personal profiles to complete strangers before deciding to interact with them. This paper tackles this open challenge by designing a suite of novel fine-grained private matching protocols. Our protocols enable two users to perform profile matching without disclosing any information about their profiles beyond the comparison result. In contrast to existing coarse-grained private matching schemes for PMSN, our protocols allow finer differentiation between PMSN users and can support a wide range of matching metrics at different privacy levels. The security and communication/computation overhead of our protocols are thoroughly analyzed and evaluated via detailed simulations.**

I. INTRODUCTION

Proximity-based mobile social networking (PMSN) becomes increasingly popular due to the explosive growth of smartphones. In particular, eMarketer estimated the US and worldwide smartphone users to be 73.3 million and 571.1 million in 2011, respectively, and almost all smartphones have WiFi and Bluetooth interfaces. PMSN refers to the social interaction among physically proximate mobile users directly through the Bluetooth/WiFi interfaces on their smartphones or other mobile devices. As a valuable complement to web-based online social networking, PMSN enables more tangible face-to-face social interactions in public places such as bars, airports, trains, and stadiums [1]. In addition, PMSN may be the only feasible social networking tool when mobile users cannot access the Internet for online social networking, e.g., due to lack of Internet access minutes or very weak signals from cellular base stations or WiFi access points.

PMSN is conducted via applications running on smartphones or other mobile devices. Such applications can be offered by small independent developers. For instance, there are currently over 50 Bluetooth/WiFi chatting applications in the Android Market for Android devices and 60 in the App Store for Apple devices. Developing advanced Bluetooth/WiFi social networking applications also has recently attracted attention from the academia [1]. Moreover, online social network providers such as Facebook and Twitter may add PMSN functionalities to their future applications for smartphones and other mobile devices.

Private (profile) matching is indispensable for fostering the wide use of PMSN. On the one hand, people normally prefer to socialize with others having similar interests or background over complete strangers. Such social reality makes *profile matching* [2] the first step towards effective PMSN, which refers to two users comparing their personal profiles before real interaction. On the other hand, people have growing privacy concerns for disclosing personal profiles to arbitrary persons in physical proximity before deciding to interact with them [2]–[5]. Although similar privacy concerns also exist in online social networking, preserving users' profile privacy is more urgent in PMSN, as attackers can directly associate obtained personal profiles with real persons nearby and then launch more targeted attacks. This situation leads to a circular dependency between personal-profile exchange and engagement in PMSN and thus necessitates *private matching*, in which two users to compare their personal profiles without disclosing them to each other.

Some elegant schemes such as [2]–[4] have recently been proposed to enable coarse-grained private matching for PMSN. Common to these schemes is the implicit assumption that each user's personal profile consists of multiple attributes chosen from a public set of attributes, which can be various interests [2], friends [3], or disease symptoms [4] in different contexts. Private matching is then converted into Private Set Intersection (PSI) [6], [7] or Private Set Intersection Cardinality (PSI-CA) [8], [9], whereby two mutually mistrusting parties, each holding a private data set, jointly compute the intersection [6], [7] or the intersection cardinality [8], [9] of the two sets without leaking any additional information to either party. These schemes [2]–[4] can enable only coarse-grained private matching and are unable to further differentiate users with the same attribute(s). For example, Alice, Bob, and Charlie all like watching movies and thus have "movie" as an attribute of their respective profile. Alice and Bob, however, both go to the cinema twice a week, while Charlie does so once every two weeks. If Alice can interact with only one of Bob and Charlie, e.g., due to time constraints, Bob is obviously a better choice. Under the existing schemes [2]–[4], however, Bob and Charlie appear the same to Alice. To solve this problem and thus

978-1-4673-0775-8/12/$31.00 ©2012 IEEE

further enhance the usability of PMSN calls for *fine-grained private matching*.

A natural first step towards fine-grained private matching for PMSN is to use fine-grained personal profiles. The basic idea is to associate a user-specific numerical value with every attribute. For example, assume that every attribute corresponds to a different interest such as movie, sports, and cooking. The first time every user uses the PMSN application, he is prompted to create his profile by assigning a value to every attribute in the public attribute set defined by the PMSN application. Every attribute value is an integer in [0, 10] and indicates the level of interest from no interest (0) to extremely high interest (10). Every personal profile is then defined as a set of attribute values, each corresponding to a unique attribute in the public attribute set.

Fine-grained personal profiles have significant advantages over traditional coarse-grained ones comprising only interested attributes from a public attribute set. First, fine-grained personal profiles enable finer differentiation among the users having different levels of interest in the same attribute. Continue with the previous example. Alice now can choose Bob over Charlie, as she and Bob have closer attribute values for "movie." In addition, fine-grained personal profiles enable personalized profile matching in the sense that two users can select the same agreed-upon metric from a set of candidate metrics to measure the similarity between their personal profiles or even different metrics according to their individual needs. The accompanying challenge is, however, how to ensure the privacy of fine-grained profile matching, which cannot be solved by existing solutions [2]–[4].

This paper explores fine-grained private (profile) matching to foster the wide use of PMSN. Our main contributions can be summarized as follows.

- We motivate the requirement for and formulate the problem of fine-grained private (profile) matching for PMSN for the first time in the literature.
- We propose the notion of fine-grained personal profiles and a corresponding suite of novel private-matching protocols for different metrics measuring profile similarity. Our first three protocols are for the ℓ_1 distance, which is the sum of absolute difference in each attribute. We also propose a threshold-based protocol based on the ℓ_1 distance, in which two users can determine whether the ℓ_1 distance between their profiles is smaller than some personally chosen threshold. We finally extend the third protocol to be a threshold-based protocol based on the MAX distance, which is the maximum absolute difference among all attributes.
- We provide thorough security analysis and performance evaluation of our proposed protocols and demonstrate their efficacy and efficiency under practical settings.

II. Problem Formulation and Cryptographic Tool

A. Proximity-based Mobile Social Networking (PMSN)

We assume that each user carries a smartphone or some other mobile device with the same PMSN application installed. The PMSN application can be developed by small independent developers or offered by online social network service providers like Facebook as a function module of their applications built for mobile devices. More advanced PMSN applications have also been developed by the academia [1]. For convenience only, we shall not differentiate a user from his mobile device later.

A PMSN session involves two users and consists of three phases. First, two users need discover each other in the neighbor-discovery phase. Second, they need compare their personal profiles in the matching phase. Last, two matching users enter the interaction phase for real information exchange. Our work is concerned with the first and second phases.

The PMSN application uses fine-grained personal profiles for fine-grained matching. In particular, the application developer defines a public attribute set consisting of d attributes $\{A_1, \dots, A_d\}$, where d may range from several tens to several hundreds depending on specific PMSN applications. The attributes may have different meanings in different contexts, such as interests [2], friends [3], or disease symptoms [4]. For easier illustration, we hereafter assume that each attribute corresponds to a personal interest such as movie, sports, and cooking. To create a fine-grained personal profile, every user selects an integer $u_i \in [0, \gamma-1]$ to indicate his level of interest in A_i (for all $i \in [1, d]$) the first time he uses the PMSN application. As a fixed system parameter, γ could be a small integer, say 5 or 10, which may be sufficient to differentiate user's interest level. The higher u_i, the more interest the user has in A_i, and vice versa. More specifically, 0 and $\gamma-1$ mean no interest and extremely high interest, respectively. Every personal profile is then defined as a vector $\langle u_1, \dots, u_d \rangle$. The user can also modify his profile later on as needed.

B. Problem Statement: Fine-grained Private Matching in PMSN

We consider Alice with profile $\mathbf{u} = \langle u_1, \dots, u_d \rangle$ and Bob with profile $\mathbf{v} = \langle v_1, \dots, v_d \rangle$ as two exemplary users of the same PMSN application from here on. Assume that Alice wants to find someone to chat with, e.g., when waiting for the flight to depart. As the first step (Neighbor Discovery), she broadcasts a chatting request via the PMSN application on her smartphone to discover proximate users of the same PMSN application. Suppose that she receives multiple responses including one from Bob who may also simultaneously respond to other persons. Due to time constraints or other reasons, both Alice and Bob can only interact with one stranger whose profile best matches hers or his. The next step (Profile Matching) is thus for Alice (or Bob) to compare her (or his) profile with those of others who responded to her (or whom he responded to). Our subsequent discussion will focus on the profile-matching process between Alice and Bob for the sake of simplicity. As in [5], we assume that the PMSN application is completely distributed and does not involve any third party.

Alice and Bob are both assumed to have privacy concerns about disclosing their personal profiles to complete strangers, so a privacy-preserving matching protocol is needed. In particular, let $\mathcal{F}$ denote a set of candidate matching metrics defined by the PMSN application developer, where each $f \in \mathcal{F}$ is a function over two personal profiles that measures their similarity. Our private-matching protocols allow Alice and Bob to either negotiate one common metric from $\mathcal{F}$ or choose different metrics according to their individual needs. We shall focus on the latter more general case henceforth, in which private matching can be viewed as two independent protocol executions, with each user initiating the protocol once according to her/his chosen metric. Assume that Alice chooses a matching metric $f \in \mathcal{F}$ and runs the privacy-matching protocol with Bob to compute $f(\mathbf{u}, \mathbf{v})$. According to the amount of information disclosed during the protocol execution, we define the following three privacy levels from Alice's viewpoint, which can also be equivalently defined from Bob's viewpoint for his chosen matching metric.

Definition 1. Level-I privacy: *When the protocol ends, Alice only learns $f(\mathbf{u}, \mathbf{v})$, and Bob only learns f.*

Definition 2. Level-II privacy: *When the protocol ends, Alice only learns $f(\mathbf{u}, \mathbf{v})$, and Bob learns nothing.*

Definition 3. Level-III privacy: *When the protocol ends, Alice only learns if $f(\mathbf{u}, \mathbf{v}) < \tau_A$ holds for some threshold τ_A of her own choice without learning $f(\mathbf{u}, \mathbf{v})$, and Bob learns nothing.*

For all three privacy levels, neither Alice nor Bob learns the other's personal profile. With level-I privacy, although Bob cannot learn $f(\mathbf{u}, \mathbf{v})$, he learns the matching metric f chosen by Alice. In contrast to level-I privacy, level-II privacy additionally requires that Bob learn nothing other than $f \in \mathcal{F}$. Finally, level-III privacy discloses the least amount of information by also hiding $f(\mathbf{u}, \mathbf{v})$ from Alice. We will introduce a suite of private-matching protocols satisfying one of the three privacy levels. Besides privacy guarantees, other design objectives include small communication and computation overhead, which can translate into the total energy consumption and matching time and thus are crucial for resource-constrained mobile devices and the usability of PMSN.

C. Cryptographic Tool: Paillier Cryptosystem

Our protocols rely on the Paillier cryptosystem [10], and we assume that every PMSN user has a unique Paillier public/private key pair which can be generated via a function module of the PMSN application. How the keys are generated and used for encryption and decryption are briefed as follows to help illustrate and understand our protocols.

- **Key generation.** An entity chooses two primes p and q and compute $N = pq$ and $\lambda = \text{lcm}(p-1, q-1)$. It then selects a random $g \in \mathbb{Z}^*_{N^2}$ such that $\gcd(\mathsf{L}(g^\lambda \mod N^2), N) = 1$, where $\mathsf{L}(x) = (x-1)/N$. The entity's Paillier public and private keys are $\langle N, g\rangle$ and λ, respectively.
- **Encryption.** Let $m \in \mathbb{Z}_N$ be a plaintext and $r \in \mathbb{Z}_N$ be a random number. The ciphertext is given by
$$\mathsf{E}(m \mod N, r \mod N) = g^m r^N \mod N^2 \;, \quad (1)$$
where $\mathsf{E}(\cdot)$ denotes the Paillier encryption operation on two integers modulo N. To simplify our expressions, we shall hereafter omit the modular notation inside $\mathsf{E}(\cdot)$.
- **Decryption.** Given a ciphertext $c \in \mathbb{Z}_{N^2}$, the corresponding plaintext can be derived as
$$\mathsf{D}(c) = \frac{\mathsf{L}(c^\lambda \mod N^2)}{\mathsf{L}(g^\lambda \mod N^2)} \mod N \;, \quad (2)$$
where $\mathsf{D}(\cdot)$ denotes the Paillier decryption operation.

The Paillier's cryptosystem has two very useful properties.

- **Homomorphic.** For any $m_1, m_2, r_1, r_2 \in \mathbb{Z}_N$, we have
$$\mathsf{E}(m_1, r_1)\mathsf{E}(m_2, r_2) = \mathsf{E}(m_1 + m_2, r_1 r_2) \mod N^2,$$
$$\mathsf{E}^{m_2}(m_1, r_1) = \mathsf{E}(m_1 m_2, r_1^{m_2}) \mod N^2 \;.$$
- **Self-blinding.**
$$\mathsf{E}(m_1, r_1) r_2^N \mod N^2 = \mathsf{E}(m_1, r_1 r_2) \;,$$
which implies that any ciphertext can be changed to another without knowing the plaintext.

The Paillier cryptosystem is semantically secure for sufficiently large N and g. To facilitate our illustrations, we assume that N and g are of 1024 and 160 bits, respectively, for sufficient semantical security [5]. Under this assumption, a public key $\langle N, g\rangle$ is of 1184 bits, a ciphertext is of $2\log_2 N$=2048 bits, a Paillier encryption needs two 1024-bit exponentiations and one 2048-bit multiplication, and a Paillier decryption costs essentially one 2048-bit exponentiation.

III. Fine-grained Private Matching Protocols

In this section, we present three private-matching protocols to support different matching metrics and offer different levels of privacy. In particular, Protocol 1 is for the ℓ_1-distance matching metric and can offer level-I privacy, Protocol 2 supports a family of additively separable matching metrics and can offer level-II privacy, and Protocol 3 is an enhancement of Protocol 2 for supporting level-III privacy.

A complete matching process involves Alice with profile $\mathbf{u} = \langle u_1, \ldots, u_d\rangle$ and Bob with profile $\mathbf{v} = \langle v_1, \ldots, v_d\rangle$, each running an independent instance of the same or even different private-matching protocol. Let f denote any matching metric supported by Protocols 1 to 3. The larger $f(\mathbf{u}, \mathbf{v})$, the less similar $\mathbf{u}$ and $\mathbf{v}$, and vice versa. We can thus consider $f(\mathbf{u}, \mathbf{v})$ some kind of distance between $\mathbf{u}$ and $\mathbf{v}$. Assume that Alice has a threshold τ_A and will accept Bob if $f(\mathbf{u}, \mathbf{v}) < \tau_A$. Similarly, Bob has a threshold τ_B and will accept Alice if $f(\mathbf{u}, \mathbf{v}) < \tau_B$. If both accept each other, they can start real information exchange. Our subsequent protocol illustrations and analysis will be from Alice's viewpoint, which can be similarly done from Bob's viewpoint. We assume that Alice has a Paillier public key $\langle N, g\rangle$ and the corresponding private key λ, which are generated as in Section II-C. A practical security protocol often involves some routines such as using timestamps to mitigate replay attacks and message authentication codes for integrity protection. To focus on explaining our key ideas, we will neglect such security routines in protocol illustrations.

A. Protocol 1 for Level-I Privacy

Protocol 1 is designed for the ℓ_1 distance as the matching metric. Recall that every personal profile is a vector of dimension d. As probably the most straightforward matching metric, the ℓ_1 distance (also called the Manhattan distance) is computed by summing the absolute value of the element-wise subtraction of two profiles and is a special case of the more general ℓ_α distance defined as

$$\ell_\alpha(\mathbf{u},\mathbf{v}) = (\sum_{i=1}^{d} |v_i - u_i|^\alpha)^{\frac{1}{\alpha}} \ , \tag{3}$$

where $\alpha \geq 1$. When $\alpha = 1$, we have $\ell_1(\mathbf{u},\mathbf{v}) = \sum_{i=1}^{d} |v_i - u_i|$. The ℓ_1 distance evaluates the overall absolute difference between two personal profiles.

Protocol 1 is designed to offer level-I privacy from Alice's viewpoint with regard to Bob. It is a nontrivial adaptation from the protocol in [13] with significantly lower computation overhead to be shown shortly. The basic idea is to first convert $\ell_1(\mathbf{u},\mathbf{v})$ into the ℓ_2 distance between the unary representations of $\mathbf{u}$ and $\mathbf{v}$ and then compute the ℓ_2 distance using a secure dot-product protocol.

In particular, for all $x \in [0, \gamma-1]$, we define a binary vector $h(x) = \langle x_1, \ldots, x_{\gamma-1}\rangle$, where x_i is equal to one for $1 \leq i \leq x$ and zero for $x < i \leq \gamma - 1$. We also abuse the notation by defining another binary vector $\hat{\mathbf{u}} = h(\mathbf{u}) = \langle h(u_1), \ldots, h(u_d)\rangle = \langle \hat{u}_1, \ldots, \hat{u}_{(\gamma-1)d}\rangle$ and $\hat{\mathbf{v}} = h(\mathbf{v}) = \langle h(v_1), \ldots, h(v_d)\rangle = \langle \hat{v}_1, \ldots, \hat{v}_{(\gamma-1)d}\rangle$. It follows that

$$\begin{aligned}\ell_1(\mathbf{u},\mathbf{v}) &= \sum_{i=1}^{d} |u_i - v_i| = \sum_{i=1}^{(\gamma-1)d} |\hat{u}_i - \hat{v}_i| \\ &= \sum_{i=1}^{(\gamma-1)d} |\hat{u}_i - \hat{v}_i|^2 = \ell_2^2(\hat{\mathbf{u}},\hat{\mathbf{v}}) \ .\end{aligned} \tag{4}$$

The correctness of the above equation is straightforward. We can further note that

$$\begin{aligned}\ell_2^2(\hat{\mathbf{u}},\hat{\mathbf{v}}) &= \sum_{i=1}^{(\gamma-1)d} |\hat{u}_i - \hat{v}_i|^2 \\ &= \sum_{i=1}^{(\gamma-1)d} \hat{u}_i^2 - 2\sum_{i=1}^{(\gamma-1)d} \hat{u}_i\hat{v}_i + \sum_{i=1}^{(\gamma-1)d} \hat{v}_i^2 \\ &= \sum_{i=1}^{(\gamma-1)d} \hat{u}_i^2 - 2\hat{\mathbf{u}}\cdot\hat{\mathbf{v}} + \sum_{i=1}^{(\gamma-1)d} \hat{v}_i^2 \ .\end{aligned} \tag{5}$$

Since Alice and Bob know $\sum_{i=1}^{(\gamma-1)d} \hat{u}_i^2$ and $\sum_{i=1}^{(\gamma-1)d} \hat{v}_i^2$, respectively, we just need a secure dot-product protocol for Bob to compute $\hat{\mathbf{u}}\cdot\hat{\mathbf{v}}$ without knowing Alice's profile $\mathbf{u}$ or disclosing his profile $\mathbf{v}$ to Alice. Subsequently, Bob can return $-2\hat{\mathbf{u}}\cdot\hat{\mathbf{v}} + \sum_{i=1}^{(\gamma-1)d} \hat{v}_i^2$ for Alice to finish computing $\ell_2^2(\hat{\mathbf{u}},\hat{\mathbf{v}})$ and thus $\ell_1(\mathbf{u},\mathbf{v})$.

Protocol Details

The detailed operations of Protocol 1 are as follows.

1. Alice does the following in sequence.
   a. Construct a vector $\hat{\mathbf{u}} = h(\mathbf{u}) = (h(u_1), \ldots, h(u_d)) = (\hat{u}_1, \ldots, \hat{u}_{(\gamma-1)d})$, where $\hat{u}_j$ is equal to one for every $j \in \mathcal{J}_\mathbf{u} = \{j|(i-1)(\gamma-1) < j \leq (i-1)(\gamma-1) + u_i, 1 \leq i \leq d\}$ and zero otherwise.
   b. Choose a distinct $r_j \in \mathbb{Z}_N$ and compute $\mathsf{E}(\hat{u}_j, r_j)$ for every $j \in [1, (\gamma-1)d]$ using her public key.
   c. Send $\{\mathsf{E}(\hat{u}_j, r_j)\}_{j=1}^{(\gamma-1)d}$ and her public key to Bob.
2. Bob does the following after receiving Alice's message.
   a. Construct a vector $\hat{\mathbf{v}} = h(\mathbf{v}) = (h(v_1), \ldots, h(v_d)) = (\hat{v}_1, \ldots, \hat{v}_{(\gamma-1)d})$, where $\hat{v}_j$ is equal to one for every $j \in \mathcal{J}_\mathbf{v} = \{j|(i-1)(\gamma-1) < j \leq (i-1)(\gamma-1) + v_i, 1 \leq i \leq d\}$ and zero otherwise.
   b. Compute
   $$\begin{aligned}\mathsf{E}(\hat{\mathbf{u}}\cdot\hat{\mathbf{v}}, \prod_{j\in\mathcal{J}_\mathbf{v}} r_j) &= \mathsf{E}(\sum_{j\in\mathcal{J}_\mathbf{v}} \hat{u}_j, \prod_{j\in\mathcal{J}_\mathbf{v}} r_j) \\ &= \prod_{j\in\mathcal{J}_\mathbf{v}} \mathsf{E}(\hat{u}_j, r_j) \mod N^2 \ .\end{aligned} \tag{6}$$
   c. Compute $\mathsf{E}((N-2)\hat{\mathbf{u}}\cdot\hat{\mathbf{v}}, s) = \mathsf{E}^{N-2}(\hat{\mathbf{u}}\cdot\hat{\mathbf{v}}, \prod_{j\in\mathcal{J}_\mathbf{v}} r_j) \mod N^2$, where $s = (\prod_{j\in\mathcal{J}_\mathbf{v}} r_j)^{N-2} \mod N$.
   d. Compute $\mathsf{E}(\sum_{j=1}^{d(\gamma-1)} \hat{v}_j^2, r)$ with a random $r \in \mathbb{Z}_N$.
   e. Compute
   $$\begin{aligned}&\mathsf{E}(\sum_{j=1}^{d(\gamma-1)} \hat{v}_j^2 - 2\hat{\mathbf{u}}\cdot\hat{\mathbf{v}}, rs) \\ &= \mathsf{E}(\sum_{j=1}^{d(\gamma-1)} \hat{v}_j^2 + (N-2)\hat{\mathbf{u}}\cdot\hat{\mathbf{v}}, rs) \\ &= \mathsf{E}(\sum_{j=1}^{d(\gamma-1)} \hat{v}_j^2, r)\cdot\mathsf{E}((N-2)\hat{\mathbf{u}}\cdot\hat{\mathbf{v}}, s) \mod N^2 \ ,\end{aligned} \tag{7}$$
   and send it back to Alice. Note that the first equality sign is because $\hat{v}_j^2 - 2\hat{\mathbf{u}}\cdot\hat{\mathbf{v}} = \hat{v}_j^2 + (N-2)\hat{\mathbf{u}}\cdot\hat{\mathbf{v}} \mod N$.
3. Alice decrypts $\mathsf{E}(\sum_{j=1}^{d(\gamma-1)} \hat{v}_j^2 - 2\hat{\mathbf{u}}\cdot\hat{\mathbf{v}}, rs)$ using her private key to get $\sum_{j=1}^{d(\gamma-1)} \hat{v}_j^2 - 2\hat{\mathbf{u}}\cdot\hat{\mathbf{v}}$ and computes
$$\ell_1(\mathbf{u},\mathbf{v}) = \sum_{j=1}^{d(\gamma-1)} \hat{v}_j^2 - 2\hat{\mathbf{u}}\cdot\hat{\mathbf{v}} + \sum_{j=1}^{d(\gamma-1)} \hat{u}_j^2. \tag{8}$$

Protocol Analysis

We now analyze the privacy provision of Protocol 1 and the related computation and communication overhead.

Theorem 1. *Protocol 1 ensures level-I privacy if the Paillier cryptosystem is semantically secure.*

Proof: Bob receives and operates only on ciphertexts $\{\mathsf{E}(\hat{u}_1, r_1)\}_{j=1}^{(\gamma-1)d}$ and does not know Alice's private key. Since the Paillier cryptosystem is semantically secure, computationally bounded Bob cannot decrypt the ciphertexts to learn anything about Alice's profile $\mathbf{u}$. As to Alice, she only gets $\sum_{j=1}^{d(\gamma-1)} \hat{v}_j^2 - 2\hat{\mathbf{u}} \cdot \hat{\mathbf{v}}$ from Bob, which equals to $\ell_1(\mathbf{u}, \mathbf{v}) - \sum_{j=1}^{d(\gamma-1)} \hat{u}_j^2$. Since Alice knows $\sum_{j=1}^{d(\gamma-1)} \hat{u}_j^2$ herself, she cannot learn any information beyond what can be inferred from the result $\ell_1(\mathbf{u}, \mathbf{v})$. ■

The computation overhead incurred by Protocol 1 is mainly related to modular exponentiations and multiplications. In particular, Alice need perform $(\gamma - 1)d$ Paillier encryptions in Step 1.a, each costing two 1024-bit exponentiations and one 2048-bit multiplication according to Eq. (1). Note that Alice can preselect many random numbers and precompute the corresponding ciphertexts in an offline manner to reduce the online matching time. [1] In addition, Alice need perform one Paillier decryption in Step 3, which is essentially a 2048-bit exponentiation. As for Bob, he need perform $\sum_{i=1}^{d} v_i - 1$ 2048-bit multiplications in Step 2.b, one 2048-bit exponentiation in Step 2.c, two 1024-bit exponentiations and one 2048-bit multiplication (i.e., one Paillier encryption) in Step 2.d, and one 2048-bit multiplication in Step 2.e. Considering Alice and Bob together, we can approximate the online computation cost of Protocol 1 to be $\sum_{i=1}^{d} v_i + 1$ 2048-bit multiplications, two 2048-bit exponentiations, and two 1024-bit exponentiations. In contrast, a direct application of the secure dot-protocol in [13] will require Bob to perform totally $(\gamma - 1)d - 1$ more 2048-bit exponentiations in Steps 2.b and 2.c.

The communication overhead incurred by Protocol 1 involves Alice sending her public key $\langle N, g\rangle$ and $(\gamma - 1)d$ ciphertexts in Step 1.c and Bob returning one ciphertext in Step 2.e. Since a public key and a ciphertext are of 1184 and 2048 bits, respectively, the total net communication cost of Protocol 1 is of $2048(\gamma-1)d + 3232$ bits without considering message headers and other fields.

B. Protocol 2 for Level-II Privacy

We now introduce Protocol 2 which can satisfy level-II privacy. In contrast to Protocol 1 working only for the ℓ_1 distance, Protocol 2 can apply to a family of additively separable matching metrics and also hide the matching metric chosen by one user from the other. The secrecy of a user's selected matching metric can help prevent an attacker from generating better tailored profiles to deceive the victim user into a successful matching.

To illustrate Protocol 2, we first introduce the definition of *additively separable* functions as follows.

Definition 4. *A function $f(\mathbf{u}, \mathbf{v})$ is additively separable if it can be written as $f(\mathbf{u}, \mathbf{v}) = \sum_{i=1}^{d} f_i(u_i, v_i)$ for some functions $f_1(\cdot), \ldots, f_n(\cdot)$.*

Many common matching metrics are additively separable. For example, the ℓ_1 distance can be written as $\ell_1(\mathbf{u}, \mathbf{v}) = \sum_{i=1}^{d} |u_i - v_i|$, the dot product is $\mathbf{u} \cdot \mathbf{v} = \sum_{i=1}^{d} u_i v_i$, and the ℓ_α norm is $\ell_\alpha^\alpha = \sum_{i=1}^{d} |u_i - v_i|^\alpha$. In addition, assuming that Alice assigns a weight w_i to attribute i, we can define the weighted ℓ_1 distance as $\sum_{i=1}^{d} w_i |u_i - v_i|$ which is also additively separable.

Protocol 2 works by first converting any additively separable function into a dot-product computation. In particular, given an additively separable similarity function f of interest, Alice constructs a vector $\tilde{\mathbf{u}} = \langle \tilde{u}_1, \ldots, \tilde{u}_{\gamma d}\rangle$, where $\tilde{u}_j = f_i(u_i, k)$, $i = \lfloor (j-1)/\gamma \rfloor + 1$, and $k = (j-1) \mod \gamma$, for all $j \in [1, \gamma d]$. Assume that Bob also relies on his profile $\mathbf{v}$ to construct a binary vector $\tilde{\mathbf{v}} = (\tilde{v}_1, \ldots, \tilde{v}_{\gamma d})$, where the jth bit $\tilde{v}_j$ equals one for all $j \in \mathcal{J}'_{\mathbf{v}} = \{j | j = (i-1)\gamma + v_i + 1, 1 \le i \le d\}$ and zero otherwise. It follows that $\tilde{u}_j \tilde{v}_j = \tilde{u}_j = f_i(u_i, v_i)$ for all $j \in \mathcal{J}'_{\mathbf{v}}$ and zero otherwise. We then can easily obtain the following result.

$$\begin{aligned} f(\mathbf{u}, \mathbf{v}) &= \sum_{i=1}^{d} f_i(u_i, v_i) = \sum_{j \in \mathcal{J}'_{\mathbf{v}}} \tilde{u}_j \\ &= \sum_{j=1}^{\gamma d} \tilde{u}_j \tilde{v}_j = \tilde{\mathbf{u}} \cdot \tilde{\mathbf{v}} \end{aligned} \tag{9}$$

So we can let Alice run a secure dot-protocol protocol with Bob to obtain $\tilde{\mathbf{u}} \cdot \tilde{\mathbf{v}} = f(\mathbf{u}, \mathbf{v})$ without disclosing $\mathbf{u}$ or f.

Protocol Details

The detailed operations of Protocol 2 are as follows.

1. Alice first constructs a vector $\tilde{\mathbf{u}}$ as discussed above and then chooses a distinct random $r_j \in \mathbb{Z}_N$ to compute $\mathsf{E}(\tilde{u}_j, r_j)$ for all $j \in [1, \gamma d]$ using her public key. Finally, she sends $\{\mathsf{E}(\tilde{u}_j, r_j)\}_{j=1}^{\gamma d}$ and her public key to Bob.
2. Bob constructs a vector $\tilde{\mathbf{v}}$ as described above after receiving Alice's message. He then computes
$$\begin{aligned} \mathsf{E}(\tilde{\mathbf{u}} \cdot \tilde{\mathbf{v}}, \prod_{j \in \mathcal{J}'_{\mathbf{v}}} r_j) &= \mathsf{E}(\sum_{j \in \mathcal{J}'_{\mathbf{v}}} \hat{u}_j, \prod_{j \in \mathcal{J}'_{\mathbf{v}}} r_j) \\ &= \prod_{j \in \mathcal{J}'_{\mathbf{v}}} \mathsf{E}(\hat{u}_j, r_j) \mod N^2 , \end{aligned} \tag{10}$$
which holds due to Eq. (9) and the homogenous property of the Paillier cryptosystem. Next, he selects a random number $r_B \in \mathbb{Z}_N$ to compute
$$\mathsf{E}(\tilde{\mathbf{u}} \cdot \tilde{\mathbf{v}}, r_B \prod_{j \in \mathcal{J}'_{\mathbf{v}}} r_j) = \mathsf{E}(\tilde{\mathbf{u}} \cdot \tilde{\mathbf{v}}, \prod_{j \in \mathcal{J}'_{\mathbf{v}}} r_j) \cdot r_B^N \mod N^2 , \tag{11}$$
which holds due to the self-blinding property of the Paillier cryptosystem introduced in Section II-C. Finally, Bob returns $\mathsf{E}(\tilde{\mathbf{u}} \cdot \tilde{\mathbf{v}}, r_B \prod_{j \in \mathcal{J}'_{\mathbf{v}}} r_j)$ to Alice.
3. Alice decrypts $\mathsf{E}(\tilde{\mathbf{u}} \cdot \tilde{\mathbf{v}}, r_B \prod_{j \in \mathcal{J}'_{\mathbf{v}}} r_j)$ and finally get $\tilde{\mathbf{u}} \cdot \tilde{\mathbf{v}}$, i.e., $f(\mathbf{u}, \mathbf{v})$.

Protocol Analysis

We now analyze the privacy provision of Protocol 2 and the related computation and communication overhead.

Theorem 2. *Protocol 2 ensures level-II privacy if the Paillier cryptosystem is semantically secure.*

[1] Alice can even do such offline computations on her regular computer and then synchronize the results to her mobile device.

The proof is similar to that of Theorem 1 except the additional point that Bob does not know the matching metric f employed by Alice. It is thus omitted here for lack of space.

Similar to that of Protocol 1, Alice need perform γd Paillier encryptions in Step 1, each requiring two 1024-bit exponentiations and one 2048-bit multiplication, which can be done beforehand in an offline manner. Moreover, the total online computation overhead of Protocol 2 can be approximated by d 2048-bit multiplications, one 2048-bit exponentiation, and one 1024-bit exponentiation, and the total net communication cost of Protocol 3 can be computed as $2048(\gamma d + 1) + 1184$ bits without considering message headers and other fields.

C. Protocol 3 for Level-III Privacy

Protocol 3 is designed to offer level-III privacy. In contrast to Protocol 2, it only lets Alice know whether $f(\mathbf{u}, \mathbf{v})$ is smaller than a threshold τ_A of her own choice, while hiding $f(\mathbf{u}, \mathbf{v})$ from her. Protocol 3 is desirable if Bob does not want Alice to know the actual $f(\mathbf{u}, \mathbf{v})$ between their profiles.

Protocol 3 is based on a special trick. In particular, assuming that there are three arbitrary integers δ, δ_1, and δ_2 such that $\delta > \delta_1 > \delta_2 \geq 0$, we have $0 < (\delta_1 - \delta_2)/\delta < 1$. Since we assume $f(\mathbf{u}, \mathbf{v})$ and τ_A both to be integers, $f(\mathbf{u}, \mathbf{v}) < \tau_A$ is equivalent to $f(\mathbf{u}, \mathbf{v}) + (\delta_1 - \delta_2)/\delta < \tau_A$ and thus $\delta f(\mathbf{u}, \mathbf{v}) + \delta_1 < \delta\tau_A + \delta_2$. On the other hand, if $f(\mathbf{u}, \mathbf{v}) \geq \tau_A$, we would have $f(\mathbf{u}, \mathbf{v}) + (\delta_1 - \delta_2)/\delta > \tau_A$. According to this observation, Bob can choose random δ, δ_1, and δ_2 unknown to Alice and then send encrypted $\delta f(\mathbf{u}, \mathbf{v}) + \delta_1$ and $\delta\tau_A + \delta_2$ to Alice. After decrypting the ciphtertexts, Alice can check whether $\delta f(\mathbf{u}, \mathbf{v}) + \delta_1$ is smaller than $\delta\tau_A + \delta_2$ to learn whether $f(\mathbf{u}, \mathbf{v}) < \tau_A$.

Protocol Details

The detailed operations of Protocol 3 are as follows.

1. Alice first constructs a vector $\tilde{\mathbf{u}}$ as in Step 1 of Protocol 2. She then chooses a distinct random $r_j \in \mathbb{Z}_N$ to compute $\mathsf{E}(\tilde{u}_j, r_j)$ for all $j \in [1, \gamma d]$ and also another distinct random r_{τ_A} to compute $\mathsf{E}(\tau_A, r_{\tau_A})$. Finally, she sends $\{\mathsf{E}(\tilde{u}_j, r_j)\}_{j=1}^{\gamma d}$, $\mathsf{E}(\tau_A, r_{\tau_A})$, and her public key to Bob.
2. Bob first constructs a binary vector $\tilde{\mathbf{v}}$ whereby to compute $\mathsf{E}(\tilde{\mathbf{u}} \cdot \tilde{\mathbf{v}}, \prod_{j \in \mathcal{J}'_{\mathbf{v}}} r_j)$ (i.e., $\mathsf{E}(f(\mathbf{u}, \mathbf{v}), \prod_{j \in \mathcal{J}'_{\mathbf{v}}} r_j)$) as in Step 2 of Protocol 2. He then randomly chooses $r'_1, r'_2, \delta, \delta_1, \delta_2 \in \mathbb{Z}_N$ such that $\delta > \delta_1 > \delta_2$ to compute
$$\mathsf{E}(\delta\tilde{\mathbf{u}} \cdot \tilde{\mathbf{v}} + \delta_1, s_1) = \mathsf{E}(\tilde{\mathbf{u}} \cdot \tilde{\mathbf{v}}, \prod_{j \in \mathcal{J}'_{\mathbf{v}}} r_j)^{\delta} \mathsf{E}(\delta_1, r'_1) \mod N^2 \tag{12}$$
and
$$\mathsf{E}(\delta\tau_A + \delta_2, r'_2 r_{\tau_A}) = \mathsf{E}(\tau_A, r_{\tau_A})^{\delta} \cdot \mathsf{E}(\delta_2, r'_2) \mod N^2 , \tag{13}$$
where $s_1 = r'_1(\prod_{j \in \mathcal{J}'_{\mathbf{v}}} r_j)^{\delta} \mod N$. Both equations hold due to the homomorphic property of the Paillier cryptosystem. Finally, Bob returns $\mathsf{E}(\delta\tilde{\mathbf{u}}\tilde{\mathbf{v}} + \delta_1, s_1)$ and $\mathsf{E}(\delta\tau_A + \delta_2, r'_2 r_{\tau_A})$ to Alice.
3. Alice decrypts the ciphertexts to get $\delta\tilde{\mathbf{u}} \cdot \tilde{\mathbf{v}} + \delta_1$ and $\delta\tau_A + \delta_2$. If the former is smaller than the latter, Alice knows $f(\mathbf{u}, \mathbf{v}) < \tau_A$. Otherwise, she knows $f(\mathbf{u}, \mathbf{v}) \geq \tau_A$.

Protocol Analysis

We now analyze the privacy provision of Protocol 3 and the related computation and communication overhead.

Theorem 3. *Protocol 3 ensures level-III privacy if the Paillier cryptosystem is semantically secure.*

The proof is similar to that of Theorem 2 except the additional points that Bob does not know Alice's threshold τ_A and that Alice does know the comparison result $f(\mathbf{u}, \mathbf{v})$. It is thus omitted here for lack of space.

Similar to that of Protocol 2, Alice need perform $\gamma d + 1$ Paillier encryptions in Step 1, each requiring two 1024-bit exponentiations and one 2048-bit multiplication, which can be done beforehand in an offline manner. Moreover, the total online computation cost of Protocol 3 can be approximated by $d + 3$ 2048-bit multiplications, four 2048-bit exponentiation, and four 1024-bit exponentiations, and the total net communication cost of Protocol 3 can be computed as $2048(\gamma d + 3) + 1184$ bits without considering message headers and other fields.

IV. Extension: MAX-Distance Matching

In this section, we present another private-matching protocol based on the MAX distance. Given two personal profiles $\mathbf{u}$ and $\mathbf{v}$, the MAX distance between them is defined as follows.
$$\ell_{\max}(\mathbf{u}, \mathbf{v}) = \max\{|v_1 - u_1|, \ldots, |v_d - u_d|\} \tag{14}$$
Protocols 1 to 3 all enable a user to check whether the overall absolute difference between her and another user's profiles is below a personally chosen threshold. In contrast, Protocol 4 allows the user to check whether the maximum attribute-wise absolute difference does not exceed her personal threshold.

At the first glance, $\ell_{\max}(\mathbf{u}, \mathbf{v})$ is not additively separable, so it cannot be computed using Protocol 2 or 3. In what follows, we first show how to convert $\ell_{\max}(\mathbf{u}, \mathbf{v})$ into an additively separable function based on a concept called *similarity matching* and then present the protocol details.

A. MAX Distance as an Additively Separable Function

The conversion from $\ell_{\max}(\mathbf{u}, \mathbf{v})$ to an additively separable function relies on similarity matching defined as follows.

Definition 5. *Given two personal profiles $\mathbf{u} = \langle u_1, \ldots, u_d\rangle$ and $\mathbf{v} = \langle v_1, \ldots, v_d\rangle$, their ith attributes are considered **similar** if $|u_i - v_i| \leq \tau$ for a specific threshold τ.*

Definition 6. *The **similarity score** of $\mathbf{u}$ and $\mathbf{v}$, denoted by $\Phi(\mathbf{u}, \mathbf{v}, \tau)$, is the total number of similar attributes, i.e.,*
$$\Phi(\mathbf{u}, \mathbf{v}, \tau) = \sum_{i=1}^{d} \phi(u_i, v_i, \tau) , \tag{15}$$
where
$$\phi(u_i, v_i, \tau) = \begin{cases} 1 & \text{if } |u_i - v_i| \leq \tau, \\ 0 & \text{otherwise} . \end{cases}$$

The similarity score has three essential properties. First, it is additively separable, implying that Alice can run Protocol 2 with Bob to compute $\Phi(\mathbf{u}, \mathbf{v}, \tau)$ or Protocol 3 to check whether $\Phi(\mathbf{u}, \mathbf{v}, \tau) < \tau_A$. Second, it is directly affected by the value of τ. In particular, the larger τ, the higher $\Phi(\mathbf{u}, \mathbf{v}, \tau)$, and vice versa. Last, it relates to $\ell_{\max}(\mathbf{u}, \mathbf{v})$ based on the following theorem.

Theorem 4. *For all $\tau \geq \ell_{\max}(\mathbf{u}, \mathbf{v})$, we have $\Phi(\mathbf{u}, \mathbf{v}, \tau) = d$; likewise, for all $\tau < \ell_{\max}(\mathbf{u}, \mathbf{v})$, we have $\Phi(\mathbf{u}, \mathbf{v}, \tau) < d$.*

Proof: By the definition of the MAX distance, we have $|u_i - v_i| \leq \ell_{\max}(\mathbf{u}, \mathbf{v})$ for all $1 \leq i \leq d$. It follows that $\phi(u_i, v_i, \tau) = 1$ for all $1 \leq i \leq d$ if $\tau \geq \ell_{\max}(\mathbf{u}, \mathbf{v})$. Therefore, we have $\mathbf{s}(\mathbf{u}, \mathbf{v}, \tau) = d$ for all $\tau \geq \ell_{\max}(\mathbf{u}, \mathbf{v})$. Similarly, by the definition of MAX distance, there exists $k \in [1, d]$ such that $|u_k - v_k| = \ell_{\max}(\mathbf{u}, \mathbf{v})$. It follows that $\phi(u_k, v_k, \tau) = 0$, so we have $\Phi(\mathbf{u}, \mathbf{v}, \tau) < d$ for all $\tau < \ell_{\max}(\mathbf{u}, \mathbf{v})$. ∎

B. Protocol 4: MAX-Distance Matching for Level-III Privacy

Protocol 4 depends on Protocol 3 for level-III privacy. Let $\tau_{\max}$ to denote Alice's MAX-distance threshold kept secret from Bob. According to Theorem 4, checking whether $\ell_{\max}(\mathbf{u}, \mathbf{v}) < \tau_{\max}$ is equivalent to checking whether $\Phi(\mathbf{u}, \mathbf{v}, \tau_{\max}) = d$.

Protocol Details

1. Alice first constructs a vector $\tilde{\mathbf{u}} = \langle \tilde{u}_1, \ldots, \tilde{u}_{\gamma d} \rangle$, where $\tilde{u}_j = \phi_i(u_i, k, \tau_{\max})$, $i = \lfloor j/\gamma \rfloor + 1$, and $k = (j-1) \bmod \gamma$ for all $j \in [1, \gamma d]$. She then chooses a random $r_{\max} \in \mathbb{Z}_N$ to compute $\mathsf{E}(d, r_{\max})$ and a distinct random $r_j \in \mathbb{Z}_N$ to compute $\mathsf{E}(\tilde{u}_j, r_j)$ for all $j \in [1, \gamma d]$. Finally, she sends $\{\mathsf{E}(\tilde{u}_j, r_j)\}_{j=1}^{\gamma d}$, $\mathsf{E}(d, r_{\tau_{\max}})$, and her public key to Bob.
2. Bob performs almost the same operations as in Step 2 of Protocol 2 (except replacing r_A by $r_{\max}$) and returns $\mathsf{E}(\delta\tilde{\mathbf{u}} \cdot \tilde{\mathbf{v}} + \delta_1, s_1)$ and $\mathsf{E}(\delta d + \delta_2, r_2' r_{\tau_{\max}})$ to Alice. As in Protocol 2, we have $\tilde{\mathbf{u}} \cdot \tilde{\mathbf{v}} = \Phi(\mathbf{u}, \mathbf{v}, \tau_{\max})$.
3. Alice does the same as in Step 3 of Protocol 3 to check whether $\delta\tilde{\mathbf{u}} \cdot \tilde{\mathbf{v}} + \delta_1 < \delta d + \delta_2$. If so, she learns $\tilde{\mathbf{u}} \cdot \tilde{\mathbf{v}} < d$ (i.e., $\Phi(\mathbf{u}, \mathbf{v}, \tau_{\max}) < d$) and thus $\ell_{\max}(\mathbf{u}, \mathbf{v}) > \tau_{\max}$. Otherwise, Alice knows $\ell_{\max}(\mathbf{u}, \mathbf{v}) \leq \tau_{\max}$.

Since Protocol 4 is a special case of Protocol 3 and thus can also ensure level-III privacy with the same communication and computation overhead as that of Protocol 3.

V. Performance Evaluation

In this section, we evaluate the communication and computation overhead as well as overall execution time of our protocols in contrast to previous work.

Since previous work [2]–[5] on private matching for PMSN is not applicable to fine-grained private matching addressed by our protocols, we only compare our work with RSV and DA. RSV refers to the ℓ_1 distance protocol in [13] and can satisfy level-I privacy, while DV refers to the protocol in [14] for computing a general additively separable function based on commutative encryption and can offer level-II privacy. We are not aware of any existing work that can offer level-III privacy as our Protocol 3. Due to space limitations, we omit the rather straightforward derivation process for the communication and computation costs of RSV and DA and refer interested readers to [13] and [14] for details.

Table I summarizes the theoretical performance of Protocols 1∼4, RSV, and DA, where $\mathsf{mul}_1, \mathsf{mul}_2, \mathsf{exp}_1$, and exp_2 denote one 1024-bit multiplication, 2048-bit multiplication, 1024-bit exponentiation, and 2048-bit exponentiation, respectively. It is clear that all our protocols incur significantly lower online computation overhead than RSV and DA with similar communication overhead.

We further use simulations to demonstrate the advantages of our protocols over RSV and DA as well as their feasibility on smartphones in what follows.

A. Simulation Setting

We follow the evaluation methodology in [2]. In particular, we assume that Alice and Bob both use a mobile device with a 400 MHz CPU, e.g., Nokia N810 announced in 2007. According to the benchmark test results in [15], it takes about 8×10^{-5} ms, 40 ms, 2.4×10^{-4} ms, and 0.25 seconds to perform one 1024-bit multiplication, one 1024-bit exponentiation, one 2048-bit multiplication, and one 2048-bit exponentiation, respectively. Since most contemporary mobile devices have a much faster CPU,[2] our simulation results here only underestimate the performance of our protocols. As in [5], we also assume that two mobile devices communicate with each other through 802.11b interface with a transmission rate 2 Mbit/s.[3]

Other simulation settings are as follows. We simulate two random profiles with each having d attributes, where every attribute value is chosen from $[0, \gamma - 1]$ uniformly at random. The performance metrics used include the offline and online computation time, the total net communication cost in bits, and the total online execution time including the online computation and communication time. For our purpose, the simulation code was written in C++, and every data point is the average of 100 runs with different random seeds. Also note that a complete matching process involves two independent executions of the same or even different private-matching protocols, initiated by Alice and Bob, respectively. For simplicity, we assume that Alice and Bob choose the same protocol and only show the results for one protocol execution. The total matching time thus should be twice the shown total online execution time. Finally, since Protocol 4 has the same communication and computation overhead as Protocol 3, its performance results are not shown for brevity.

B. Simulation Results

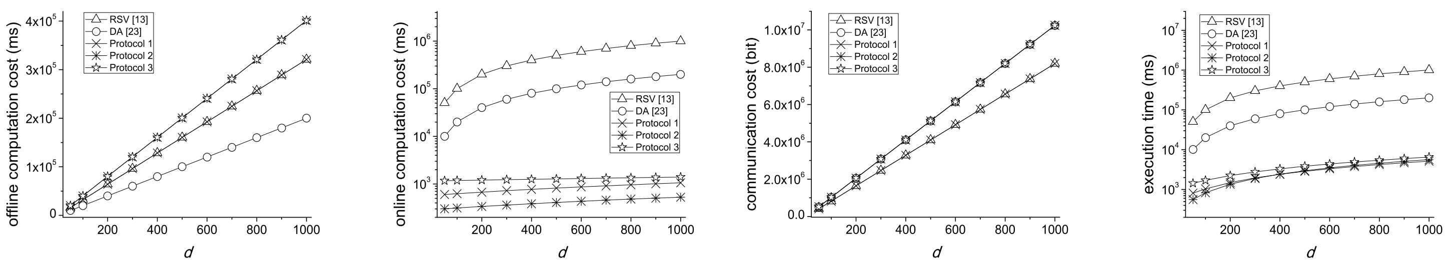
We first check the case when $\gamma = 5$ and d varies. It is not surprising to see from Fig. 1(a) that the offline computation

[2] http://www.techautos.com/2010/03/14/smartphone-processor-guide/

[3] 802.11b cards can operate at 11, 5.5, 2, and 1 Mbit/s depending on signal quality.

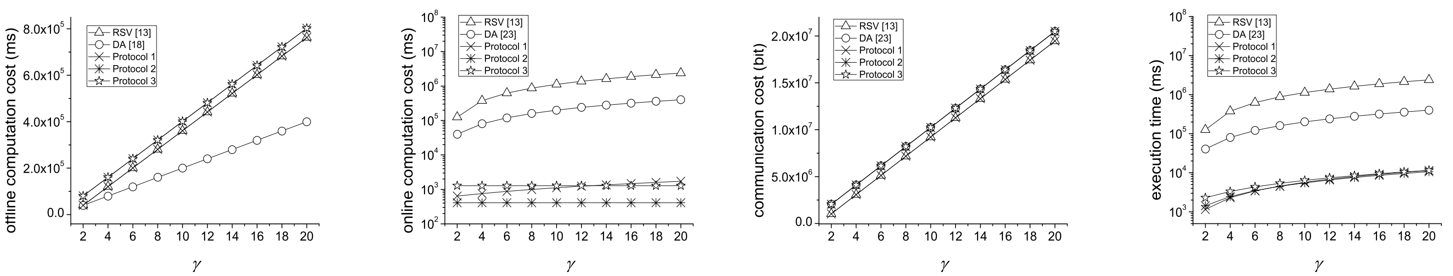
TABLE I
COMPARISON OF PRIVATE-MATCHING PROTOCOLS

Protocol	Metric	Privacy	Offline Comp.	Online Comp.	Comm. (in bits)
RSV [13]	$\ell_1(\mathbf{u}, \mathbf{v})$	Level-I	$2(\gamma-1)d\ \mathsf{exp}_1, (\gamma-1)d\ \mathsf{mul}_2$	$(\gamma-1)d+1\ \mathsf{exp}_2, 2\ \mathsf{exp}_1, (\gamma-1)d+3\ \mathsf{mul}_2$	$2048(\gamma-1)d+3232$
DA [14]	$f(\mathbf{u}, \mathbf{v})$	Level-II	$\gamma d\ \mathsf{exp}_1$	$\gamma d+2\ \mathsf{exp}_1$	$2048\gamma d+3072$
Protocol 1	$\ell_1(\mathbf{u}, \mathbf{v})$	Level-I	$2(\gamma-1)d\ \mathsf{exp}_1, (\gamma-1)d\ \mathsf{mul}_2$	$2\ \mathsf{exp}_2, 2\ \mathsf{exp}_1, \sum_{j=1}^{d} v_i+1\ \mathsf{mul}_2$	$2048(\gamma-1)d+3232$
Protocol 2	$f(\mathbf{u}, \mathbf{v})$	Level-II	$2\gamma d\ \mathsf{exp}_1, \gamma d\ \mathsf{mul}_2$	$1\ \mathsf{exp}_2, 1\ \mathsf{exp}_1, d\ \mathsf{mul}_2$	$2048\gamma d+3232$
Protocol 3	$f(\mathbf{u}, \mathbf{v}) < \tau$	Level-III	$2\gamma d+2\ \mathsf{exp}_1, \gamma d+1\ \mathsf{mul}_2$	$4\ \mathsf{exp}_2, 4\ \mathsf{exp}_1, d+3\ \mathsf{mul}_2$	$2048\gamma d+7328$
Protocol 4	$\ell_{\max}(\mathbf{u}, \mathbf{v}) < \tau$	Level-III	$2\gamma d+2\ \mathsf{exp}_1, \gamma d+1\ \mathsf{mul}_2$	$4\ \mathsf{exp}_2, 4\ \mathsf{exp}_1, d+3\ \mathsf{mul}_2$	$2048\gamma d+7328$



(a) offline computation cost (b) online computation cost (c) communication cost (d) protocol execution time

Fig. 1. Impact of the profile dimension d, where $\gamma = 5$.



(a) offline computation cost (b) online computation cost (c) communication cost (d) protocol execution latency

Fig. 2. Impact of the highest attribute value γ, where $d = 200$.

costs of all the five protocols are proportional to d. In addition, Protocols 2 and 3 incur comparable offline computation overhead higher than that of Protocol 1 and RSV which incur the same computation overhead, and the offline computation overhead incurred by DA is the lowest. The main reason is that both Protocols 2 and 3 require $\gamma d + 1$ offline Paillier encryptions, [4] while RSV and Protcol 1 require $(\gamma - 1)d$. In contrast, DA need perform γd offline commutative encryptions, each of which corresponds to one 1024-bit exponentiation and is thus more efficient than one Paillier encryption. Since users can do such offline computations on their regular computers and then synchronize the results to their mobile devices. The offline computation cost thus does not contribute to the total protocol execution time.

Fig. 1(b) shows the online computation costs of all the protocols in the $\log 10$ scale for a fixed $\gamma = 5$ and varying d. It is clear that Protocols 1 to 3 all incur much lower online computation overhead than both RSV and DA. The main reasons are that 1024-bit and 2048-bit exponentiations dominate the online computation costs of all the protocols and that Protocols 1 to 3 all require a constantly small number of modular exponentiations, while RSV and DA both require a much larger number of modular exponentiations that increases almost linearly with d.

Fig. 1(c) compares the total net communication costs of all the protocols for a fixed $\gamma = 5$ and varying d. We can see that all the protocols incur comparable communication costs which all increase almost linearly with d, which is of no surprise.

Fig. 1(d) shows the total protocol execution time for a fixed $\gamma = 5$, which comprise the online computation and communication time and is dominated by the former. Protocol 2 has the shortest execution time among Protocols 1 to 3, while Protocol 3 has the longest for achieving level-III privacy. All our protocols, however, can finish within 3.3 seconds under all simulated scenarios in contrast to the much longer execution time required by RSV and also DA. For example, when $d = 200$, RSV and DA require 201.3 and 402.9 seconds to finish, respectively, while Protocols 1 to 3 only require 1.5, 1.4, and 2.2 seconds, respectively. Recall that a complete private-matching process involves two protocol executions. Our three protocols are thus much more feasible and user-friendly solutions to private matching for PMSN.

The impact of γ on the protocol performance is shown in Fig. 2, where d is fixed to be 200. Similar results can be observed as in Fig. 1 and omitted here for space constraints.

[4]Recall that one Paillier encryption corresponds to two 1024-bit exponentiation and one 2048-bit multiplication.

VI. Related Work

As mentioned in Section I, the private matching schemes proposed in [2]–[4] aim at coarse-grained personal profiles and match two users based on a privacy-preserving computation of the intersection (cardinality) of their attribute sets. In contrast, our protocols support fine-grained personal profiles and thus much finer user differentiation, which is important for fostering the much wider use of PMSN. To our best knowledge, Dong *et al.* presented the only piece of work in [5] that does not match two users in PMSN using the intersection (cardinality) of their attribute sets. Instead, they proposed using the social proximity between two users as the matching metric, which measures the distance between their social coordinates with each being a vector precomputed by a trusted central server to represent the location of a user in an online social network. By comparison, our work does not rely on the affiliation of PMSN users with a single online social network and addresses a more general private matching problem for PMSN by supports fine-grained personal profiles and a wide spectrum of matching metrics.

Private matching for PMSN can also be viewed as special instances of secure two-party computation, which was initially introduced by Yao in [16] and later generalized to secure multi-party computation by Goldreich *et al.* [17] and many others. In secure two-party computation, two users with private inputs x and y, respectively, both want to compute some function $f(x, y)$ without any party learning information beyond what can be inferred from the result $f(x, y)$. It was shown that all secure multi-party computation problems can be solved using the general approach [17], which is nevertheless too inefficient to use in practice. The existing literature thus focused on devising more efficient solutions for specific functions.

Securely computing some function over two vectors has also been investigated in the context of privacy-preserving data mining and scientific computation. In particular, secure dot-product computation was studied in [18]–[21]. As in [5], we adopt the method in [20] as a component of our protocols and make significant contributions on relating the computation of many PMSN matching metrics to secure dot-product computation. Privacy-preserving correlation computation was studied in [22], [23] and is loosely related to our work here. In addition, Du and Atallah proposed a set of protocols for securely computing the difference between two private vectors based on different metrics [14], including the ℓ_1 distance, the ℓ_2 distance, and a more general function. Their protocols are based on relatively expensive commutative encryptions, and it is not clear how to apply their protocols to our problem here in an efficient and secure fashion. Moreover, some novel methods were proposed in [13] for securely computing the approximate ℓ_1 distance of two private vectors. As said before, our Protocol 1 is adapted from the protocols [13] but with significantly lower computation overhead.

VII. Conclusion and Future Work

In this paper, we formulated the problem of fine-grained private (profile) matching for proximity-based mobile social networking and presented a suite of novel solutions that support a variety of private-matching metrics at different privacy levels. Detailed performance analysis and evaluation confirmed the high efficiency of our protocols over prior work under practical settings. Our future work is to implement our protocols as real smartphone applications and make them available for the public.

Acknowledgement

This work was supported in part by the US National Science Foundation under grants CNS-1117462 and CNS-0844972 (CAREER). We would also like to thank anonymous reviewers for their constructive comments and helpful advice.

References

[1] 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 *ICDCS'10*, Genoa, Italy, June 2010.

[2] M. Li, N. Cao, S. Yu, and W. Lou, "FindU: Privacy-preserving personal profile matching in mobile social networks," in *INFOCOM'11*, Shanghai, China, Apr. 2011.

[3] M. von Arb, M. Bader, M. Kuhn, and R. Wattenhofer, "VENETA: Serverless friend-of-friend detection in mobile social networking," in *WIMOB'08*, Avignon, France, Oct. 2008, pp. 184–189.

[4] R. Lu, X. Lin, X. Liang, and X. Shen, "A secure handshake scheme with symptoms-matching for mhealthcare social network," *Mobile Networks and Applications*, pp. 1–12, 2010.

[5] W. Dong, V. Dave, L. Qiu, and Y. Zhang, "Secure friend discovery in mobile social networks," in *INFOCOM'11*, Shanghai, China, Apr. 2011.

[6] L. Kissner and D. Song, "Privacy-preserving set operations," in *CRYPTO'05*, Santa Barbara, CA, Aug. 2005, pp. 241–257.

[7] Q. Ye, H. Wang, and J. Pieprzyk, "Distributed private matching and set operations," in *ISPEC'08*, vol. 4991, Sydney, Australia, Apr. 2008.

[8] M. Freedman *et al.*, "Efficient private matching and set intersection," in *EUROCRYPT'04*, Interlaken, Switzerland, May 2004.

[9] E. Cristofaro and G. Tsudik, "Practical private set intersection protocols with linear complexity," in *FC'10*, Tenerife, Spain, Jan. 2010.

[10] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *EUROCRYPT'99*, Prague, Czech Republic, May 1999.

[11] A. Juels and J. Brainard, "Client puzzles: A cryptographic countermeasure against connection depletion attacks," in *NDSS'99*, San Diego, CA, Feb. 1999.

[12] S. Gollakota, N. Ahmed, N. Zeldovich, and D. Katabi, "Secure in-band wireless pairing," in *USENIX Security*, San Francisco, CA, Aug. 2011.

[13] S. Rane *et al.*, "Privacy-preserving approximation of L_1 distance for multimedia applications," in *ICME'10*, Singapore, July 2010.

[14] W. Du and M. J. Atallah, "Protocols for secure remote database access with approximate matching," in *WSPEC'00*, Athens, Greece, Nov. 2000.

[15] S. Bhatt *et al.*, "A personal mobile drm manager for smartphones," *Computers & Security*, vol. 28, no. 6, pp. 327–340, 2009.

[16] A. C. Yao, "Protocols for secure computations," in *SFCS'82*, Washington, DC, USA, Nov 1982, pp. 160–164.

[17] O. Goldreich, S. Micali, and A. Wigderson, "How to play ANY mental game," in *STOC'87*, New York, NY, 1987, pp. 218–229.

[18] W. Du and M. Atallah, "Privacy-preserving cooperative statistical analysis," in *ACSAC'01*, New Orleans, Louisiana, Dec. 2001, pp. 102–110.

[19] I. Ioannidis, A. Grama, and M. Atallah, "A secure protocol for computing dot-products in clustered and distributed environments," in *ICPP'02*, Vancouver, British Columbia, Canada, Aug. 2002, pp. 379–384.

[20] B. Goethals, S. Laur, H. Lipmaa, and T. Mielikainen, "On private scalar product computation for privacy-preserving data mining," in *ICISC'04*, vol. 3506, Seoul, Korea, Dec. 2004, pp. 23–25.

[21] M. Shaneck and Y. Kim, "Efficient cryptographic primitives for private data mining," in *HICSS'10*, Jan. 2010, pp. 1–9.

[22] H. Polat and W. Du, "Privacy-preserving collaborative filtering using randomized perturbation techniques," in *ICDM'03*, Melbourne, FL, Nov. 2003.

[23] A. Kiayias, B. Yener, and M. Yung, "Privacy-preserving information markets for computing statistical data," in *FC'09*, Berlin, Heidelberg, Feb. 2009.