Privacy-Preserving Distributed Profile Matching in Proximity-based Mobile Social Networks

Ming Li, *Member, IEEE,* Shucheng Yu, *Member, IEEE,* Ning Cao, *Student Member, IEEE,* and Wenjing Lou, *Senior Member, IEEE*

***Abstract*—Making new connections according to personal preferences is a crucial service in mobile social networking, where an initiating user can find matching users within physical proximity of him/her. In existing systems for such services, usually all the users directly publish their complete profiles for others to search. However, in many applications, the users' personal profiles may contain sensitive information that they do not want to make public. In this paper, we propose FindU, a set of privacy-preserving profile matching schemes for proximity-based mobile social networks. In FindU, an initiating user can find from a group of users the one whose profile best matches with his/her; to limit the risk of privacy exposure, only necessary and minimal information about the private attributes of the participating users is exchanged. Two increasing levels of user privacy are defined, with decreasing amounts of revealed profile information. Leveraging secure multi-party computation (SMC) techniques, we propose novel protocols that realize each of the user privacy levels, which can also be personalized by the users. We provide formal security proofs and performance evaluation on our schemes, and show their advantages in both security and efficiency over state-of-the-art schemes.**

I. Introduction

With the proliferation of mobile devices, mobile social networks (MSNs) are becoming an inseparable part of our lives. Leveraging networked portable devices such as smart phones and PDAs as platforms, MSN not only enables people to use their existing online social networks (OSNs) at anywhere and anytime, but also introduces a myriad of mobility-oriented applications, such as location-based services and augmented reality. Among them, an important service is to make new social connections/friends within physical proximity based on the matching of personal profiles. For example, MagnetU and E-SmallTalker [2] are MSN applications that match one with nearby people for dating or friend-making based on common interests. In such an application, a user only needs to input some (query) attributes in her profile, and the system would automatically find the persons around with similar profiles. The scopes of these applications are very broad, since people can input anything as they want, such as hobbies, phone contacts and places they have been to. The latter can even be used to find "lost connections" and "familiar strangers".

However, such systems also raise a number of privacy concerns. Let us first examine a motivating scenario. In a hospital, patients may include their illness symptoms and medications in their personal profiles in order to find similar patients, for physical or mental support. In this scenario, an initiating user (initiator) may want to find out the patient having the maximum number of identical symptoms with her, while being reluctant to disclose her sensitive illness information to the rest of the users, and the same for the users being matched with. If users' private profiles are directly exchanged with each other, it will facilitate user profiling where those information can be easily collected by a nearby user, either in an active or passive way; and those user information may be exploited in unauthorized ways. For example, a salesman from a pharmacy may submit malicious matching queries to obtain statistics on patients' medications for marketing purposes. To cope with user profiling in MSNs, it is essential to disclose minimal and necessary personal information to as few users as possible.

In fact, the ideal situation is to let the initiator and its best matching user directly and privately find out and connect to each other, without knowing anything about other users' profile attributes, while the rest of the users should also learn nothing about the two user's matching attributes. However, it is challenging to find out the matching users privately while efficiently. One may think of simply turning off the cellphone or input very few attributes, but these would interfere with the system usability. Recently, Yang *et. al.* proposed E-SmallTalker [2], a practical system for matching people's interests before initiating a small-talk. However, E-SmallTalker suffers from the dictionary attack which does not fully protect the non-match attributes between two users. Another difficulty of private matching under a MSN setting is the lack of a centralized authority. Lu *et. al.* [3] proposed a symptom matching scheme for mobile health social networks, assuming the existence of a semi-online central authority.

In this paper, we overcome the above challenges and make the following main contributions.

(1) We formulate the privacy preservation problem of profile matching in MSN. Two levels of privacy are defined along with their threat models, where the higher privacy level leaks less profile information to the adversary than the lower level.

(2) We propose two fully distributed privacy-preserving profile matching schemes, one of them being a private set-intersection (PSI) protocol and the other is a private cardinality of set-intersection (PCSI) protocol. However, solutions based on existing PSI schemes are far from efficient. We leverage secure multi-party computation (SMC) based on polynomial secret sharing, and propose several key enhancements to improve the computation and communication efficiency. Also,

The preliminary version of this paper appeared in IEEE INFOCOM 2011 [1]. Ming Li is with the Dept. of CS, Utah State University, Logan, UT 84322. Email: ming.li@usu.edu. Shucheng Yu is with the Dept. of CS, University of Arkansas at Little Rock, Email: sxyu1@ualr.edu. Ning Cao is with the Dept. of ECE, Worcester Polytechnic Institute, Worcester, MA 01609, Email: ncao@ece.wpi.edu. Wenjing Lou is with the Dept. of CS, Virginia Tech. Email: wjlou@vt.edu.

users can choose personalized privacy levels when running the same matching instance.

(3) We provide formal security proofs and extensive performance evaluation for our schemes. Our two protocols are shown to be secure under the honest-but-curious (HBC) model, with information-theoretic security (for PSI) and standard security (for PCSI), respectively. We also discuss possible extensions to prevent malicious attacks. Meanwhile, they are shown to be more efficient than previous schemes that achieve similar security guarantees under the typical settings of MSN.

II. Problem Definition

A. System Model

Our system consists of N users (parties) denoted as $P_1, ..., P_N$, each possessing a portable device. We denote the initiating party (*initiator*) as P_1. P_1 launches the matching process and its goal is to find one party that best *"matches"* with it, from the rest of the parties $P_2, ..., P_N$ which are called *candidates*. Each party P_i's profile consists of a set of attributes $\mathcal{S}_i$, which can be strings up to a certain length. P_1 defines a matching query to be a subset of $\mathcal{S}_1$, and in the following we use $\mathcal{S}_1$ to denote the query set unless specified. Also, we denote $n = |S_1|$ and $m = |S_i|, i > 1$, assuming each candidate has the same set size for simplicity. Note that, we assume that the system adopts some standard way to describe every attribute, so that two attributes are exactly the same if they are the same semantically.

There could be various definitions of "match". In this paper, we consider a popular similarity criterion, namely the intersection set size $|\mathcal{S}_1 \cap \mathcal{S}_i|$ (also used in [2]). The larger the intersection set size, the higher the similarity between two users' profiles. User P_1 can first find out her similarity with each other users via our protocols, and then will decide whether to connect with a best matching user based on their actual common attributes.

We assume devices communicate through wireless interfaces such as bluetooth or WIFI. For simplicity, we assume every participating device is in the communication range with each other. In addition, we assume that a secure communication channel has been established between each pair of users. We will discuss practical approaches for secure setup in Sec. IV-E.

We *do not* assume the existence of a trusted third party during the protocol run; all parties carry out profile matching in a completely distributed way. They may cooperate with each other, i.e., when P_1 runs the protocol with each P_i, a subset of the rest of parties would help them to compute their results. Note that, providing incentives for the users to cooperate is an important topic, and there are some existing mechanisms [4].

B. Adversary Model

In this paper, we are mainly interested in insiders who are legitimate participators of the matching protocol and try to perform *user profiling*, i.e., obtain as much personal profile information of other nearby users as possible. For example, With a user's attributes, a bad guy could correlate and identify that user via its MAC addresses or public keys. However, we cannot absolutely prevent user profiling, because at least the initiator and its best matching user will mutually learn their intersection set. Thus we focus on minimizing the amount of private information revealed in one protocol run.

The main adversary model considered in this paper is *honest-but-curious* (HBC), i.e., a participant will infer private information from protocol run but honestly follow the protocol. Although we do not specifically address the malicious attacker model [5] where an adversary may arbitrarily deviate from the protocol run, we will discuss how our protocols can be extended to achieve security in that model. The adversary may act alone or several parties may collude. We assume that the size of a coalition is smaller than a threshold t, where t is a parameter. And we assume $N \geq 2t + 1$ (honest majority) in this paper.

C. Design Goals

1) Security Goals: Since the users may have different privacy requirements and it takes different amount of efforts to achieve them, we hereby (informally) define two levels of privacy where the higher level leaks less information to the adversaries. Aformal definition will be given in Sec. V.

Definition 1 (Privacy level 1 (PL-1)): When the protocol ends, P_1 and each candidate $P_i, 2 \leq i \leq N$ mutually learn the intersection set between them: $\mathcal{I}_{1,i} = \mathcal{S}_1 \cap \mathcal{S}_i$. An adversary $\mathcal{A}$ (whose capability is defined in Sec. II-B) should learn nothing beyond what can be derived from the above outputs and its private inputs.

If we assume the adversary has unbounded computing power, PL-1 actually corresponds to unconditional security for all the parties under the HBC model. In PL-1, P_1 can obtain all candidates' intersection sets within one protocol run, which may still reveal much user information to the attacker. Therefore we define privacy level 2 in the following.

Definition 2 (Privacy level 2 (PL-2)): When the protocol ends, P_1 and each candidate $P_i, 2 \leq i \leq N$ mutually learn the size of their intersection set: $m_{1,i} = |\mathcal{S}_1 \cap \mathcal{S}_i|$. The adversary $\mathcal{A}$ should learn nothing beyond what can be derived from the above outputs and its private inputs.

In PL-2, except when $m_{1,i} = |\mathcal{S}_1|$ or $|\mathcal{S}_i|$, P_1 and each P_i both will not learn exactly which attributes are in $\mathcal{I}_{1,i}$. The adversary needs to run the protocol multiple times to obtain the same amount of information with what he can obtain under PL-1 when he assumes the role of P_1.

2) Usability and Efficiency: For profile matching in MSN, it is desirable to involve as few human interactions as possible. In this paper, a human user only needs to explicitly participate in the end of the protocol run, e.g., decide whom to connect to based on the common interests. In addition, the system design should be *lightweight and practical*, i.e., being enough efficient in computation and communication to be used in MSN. Finally, different users (especially the candidates) shall have the option to flexibly *personalize their privacy levels*.

D. Design Challenges and Related Works

It is very challenging to achieve all the design goals simultaneously, especially if we desire high level of security but are unwilling to pay the cost of high computation and communication overhead. Similar problems to ours can be found in the literature, namely two-party private set intersection (PSI) [5]–[7], private cardinality of set intersection (PCSI) [5], [6], [8]. Our privacy goals can be achieved if given multiple instances of PSI and PCSI, respectively. They are usually tackled with Secure Multi-party Computation (SMC). The general SMC techniques [9] heavily rely on cryptography, and are well-known for their inefficiency. Researchers have proposed various customized solutions for those problems; for example, based on oblivious polynomial evaluation [5], [6], [8], [10], [11] and oblivious pseudo-random functions [7], [12], [13] that are secure in the HBC model. But when applied to the problem presented here, they incur either high computation or communication cost, thus are impractical to be used in MSN.

Concurrently with our work, a secure friend discovery protocol has been proposed in [14]. Different from us, their matching is based on computing the similarity (dot product) between two users' coordinates (which is not as intuitive as the intersection of the profile attributes as ours). In addition, a centralized trusted authority is needed to provide the coordinates. In [15], a private contact discovery protocol is proposed, where contact list manipulation is prevented by distributed certification. However, for general sensitive profile attributes it is difficult to find a distributed certifier in practice, whereas our protocols are not limited in the type of attributes to share with. In [16], privacy-preserving multi-party interest sharing protocols for smartphone applications are proposed. However, their protocols rely on an online semi-trusted server, which may not be available when the users do not have connections to it (e.g., poor signals).

In this paper, we propose two fully-distributed privacy-preserving profile matching protocols, without relying on a client-server relationship nor any central server. We propose novel methods to reduce energy consumption and protocol run time, while achieving reasonable security levels. Specifically, we exploit the homomorphic properties of Shamir secret sharing to compute the intersection between user profiles privately, and due to the smaller computational domain of secret sharing, our protocols achieve higher performance and lower energy consumption for practical parameter settings of an MSN. Such a framework is also applicable to many scenarios beyond the motivating problems in this paper, for example, in patient matching in online healthcare social networks.

III. Technical Preliminaries

Notations. We give the main notations in the following. Note that, unless specified, we denote $[s]_i$ as P_i's $(t, 2t+1)$-share of secret s under Shamir secret sharing (SS) scheme, and when we mention P_i, we refer to $2 \leq i \leq N$.

Preliminaries. **Shamir secret sharing scheme (SS)**. A (t, w)-SS scheme [17] shares secret s among w parties by giving each party P_i the value $[s]_i^{t,w}$, and if any at most t parties collude they cannot gain any information about s.

Secure multiparty computation (SMC) based on SS. For addition, SS is homomorphic: let α and β be two secrets

TABLE I: Main Notations

N, t :	Number of parties, maximum number of colluders
$[s]_i^{t,w}$:	Party P_i's secret share of s (under (t, w)-SS)
$\mathcal{S}_1$, $\mathcal{S}_i$:	P_1's query attribute set, and P_i's profile attribute set
$x_j, 1 \leq j \leq n$:	P_1's query set elements, $n = \|\mathcal{S}_1\|$
$y_{ij}, 1 \leq j \leq m$:	P_i's profile set elements, $m = \|\mathcal{S}_i\|, i \in \{2, \cdots, N\}$
$\mathcal{I}_{1,i}$:	Intersection set between P_1 and P_i; $m_{1,i} = \|\mathcal{I}_{1,i}\|$
$\mathcal{F}_p$:	The finite field used; $\Bbbk = \log p$: security parameter
$\mathcal{H}()$:	A cryptographic hash function
$\xleftarrow{R}$, $\|\|$:	Random sampling from a set, concatenation
$\mathcal{P}$, P_1, P_i:	The set of all parties, the initiator and the ith party
$\mathcal{P}_i$, $\mathcal{P}_i'$:	The computing set and reconstruction set for P_i

shared using (t, w)-SS, we have $[\alpha + \beta]_i^{t,w} = [\alpha]_i^{t,w} + [\beta]_i^{t,w}$, denoted as SS-Add. However, for secure multiplication, one round of communication is needed and it is required that $w \geq 2t + 1$ [18]. Gennaro *et. al.* proposed the following efficient secure multiplication protocol [19]. Let the inputs of party P_i be $[\alpha]_i^{t,w}$ and $[\beta]_i^{t,w}$; the idea is to first locally multiply these shares, which lie on a $2t$-degree polynomial (but not random). Thus their protocol realizes randomization and degree-reduction in one round by letting each P_i pick a random t-degree polynomial and re-share $[\alpha]_i^{t,w}[\beta]_i^{t,w}$ to others:

Round 1. Each party P_i shares the value $[\alpha]_i^{t,w}[\beta]_i^{t,w}$ by choosing a t-degree random polynomial $h_i(x)$, s. t. $h_i(0) = [\alpha]_i^{t,w}[\beta]_i^{t,w}$. He sends the value $h_i(j)$ to party P_j, $1 \leq j \leq w$.

Round 2: Every party P_j computes his share of $\alpha\beta$, i.e., the value $H(j) = [\alpha\beta]_j^{t,w}$ under a t-degree random polynomial H, by locally computing the linear combination $H(j) = \sum_{i=1}^{w} \lambda_i h_i(j)$, where $\lambda_1, ..., \lambda_w$ are known constants (Lagrangian coefficients).

This protocol incurs $O(w^2\Bbbk)$ communication cost in total, and $O(w\Bbbk)$ for each party. We denote the above protocol as SS-Mul.

Additive homomorphic encryption. An additive homomorphic encryption scheme E allows one to compute $E(m_1 + m_2)$ given $E(m_1)$ and $E(m_2)$, without knowing the underlying plain texts. This is only used in our protocol for PL-2.

IV. Main Design of FindU

In this section, we first outline the idea of FindU, and then present two core designs for the PSI and PCSI protocols. Finally we address practical issues including user discovery.

A. Overview

We present two protocols that aim at realizing one level of privacy requirement each. We start with the basic scheme realizing PSI under PL-1, which is based on secure polynomial evaluation using secret sharing. At a high level, for P_1 and each P_i $(2 \leq i \leq N)$, their inputs are shared among a subset $\mathcal{P}_i$ of $2t + 1$ parties (the computing set) using $(t, 2t + 1)$-SS, based on which they cooperatively compute shares of the function $F_i(x_j) = R_{ij} \cdot f_i(x_j) + x_j$ for each $1 \leq j \leq n$, where $f_i(y)$ is the polynomial representing P_i's set, and R_{ij} is a random number jointly generated by P_1 and P_i but not known to any party. We have $x_j \in \mathcal{I}_{1,i}$ iff. $F_i(x_j) = x_j$. The values of $\{F_i(x_j)\}_{1 \leq j \leq n}$ remain in secret-shared forms between P_1 and P_i before their shares are revealed to each other. To reduce the communication complexity, we propose an enhancement that aggregates multiple multiplication and addition operations into one round during the secure polynomial evaluation computation.

For PL-2, the advanced scheme achieves efficient PCSI. The main idea is that, the parties in $\mathcal{P}_i$ first compute the $(t, 2t+1)$-shares of the function $F_i(x_j) = R_{ij} \cdot f_i(x_j)$, $1 \leq j \leq n$ securely using the basic scheme, whereas $x_j \in \mathcal{I}_{1,i}$ iff. $R_{ij} \cdot f_i(x_j) = 0$. In order to blind from P_1 the correspondence between its inputs $\{x_j\}$ $(j \in \{1, \cdots, n\})$ and the outputs $F_i(x_{j'})$ $(j' \in \{1, \cdots, n\})$, we employ a blind-and-permute (BP) method. To reduce the number of invocations of the BP protocol, we use share conversion to convert the $(t, t+1)$-shares of $\{F_i(x_j)\}_{1 \leq j \leq n}$ (held by parties in the reconstruction set $\mathcal{P}'_i$) into $(1, 2)$-shares shared between P_1 and P_i, so that only one BP invocation is needed between P_1 and each P_i.

B. The Basic Scheme

We first give two definitions that capture the idea to involve the minimum number of parties during computation.

Definition 3 (Computing set of P_i): A set of $2t+1$ parties $\mathcal{P}_i \subset \mathcal{P}$, who help P_1 and P_i to compute the shares of $F_i(x_j)$, $1 \leq j \leq n$. $\mathcal{P}_i$ includes P_1 and P_i, and the rest $2t-1$ parties are chosen as $P_{i+1}, P_{i+2}, \cdots$ with indices wrapping around.

Definition 4 (Reconstruction set of P_i): A set of $t+1$ parties $\mathcal{P}'_i \subset \mathcal{P}_i$, who will contribute the shares of $F_i(x_j)$, $1 \leq j \leq n$ to P_1 and P_i for reconstruction, $\mathcal{P}'_i$ also includes P_1 and P_i, and the rest $t-1$ parties are chosen in the same way as in the computing set.

As input, each party has a set of attributes: P_1 has $\mathcal{S}_1 = \{x_1, x_2, ..., x_n\}$ and P_i has $\mathcal{S}_i = \{y_{i1}, y_{i2}, ..., y_{im}\}$, respectively, where each element is an encoded attribute in $\mathcal{F}_p$. For example, a hash algorithm can be used for encoding. Rather than publishing the sets as they are, each P_i first generates an m-degree polynomial based on $\mathcal{S}_i$ as follows:

$$f_i(y) = (y - y_{i1}) \cdot (y - y_{i2}) \cdots (y - y_{im}) = \sum_{k=0}^{m} a_{ik} y^k, \quad (1)$$

where $\{a_{ik}\}_{0 \leq k \leq m-1}$ are coefficients. We require $a_{im} \equiv 1$ so that P_i cannot give an all-zero polynomial. The function to be computed is: $F_i(x_j) = R_{ij} \cdot f_i(x_j) + x_j$ for each $1 \leq j \leq n$, where $R_{ij} = r_{ij} r'_{ij}$, r_{ij} and r'_{ij} are random numbers generated by P_1 and P_i, respectively. In this way, if $F_i(x_j) \in \mathcal{S}_i$, $x_j \in \mathcal{I}_{1,i}$ with high probability, and if $F_i(x_j) \notin \mathcal{S}_1$ then $x_j \notin \mathcal{I}_{1,i}$.

The basic scheme consists of three phases, and Fig. 1 describes one run between two parties - P_1 and P_i. The whole protocol between P_1 and $P_2, ..., P_N$ consists of $N-1$ instances of the two-party protocol, which can be parallelized/aggregated to save time (details are shown in [1]). In the *data share distribution* phase, P_1 shares the 1 to m powers of each of its set elements, while P_i shares its private inputs among P_i's computing set. In addition, P_1 and P_i also share their n random numbers, respectively.

In the *computation* phase, the parties in $\mathcal{P}_i$ participate in secure computation of the shares of $\{F_i(x_j)\}_{1 \leq j \leq n}$. In particular, to evaluate $f_i(x_j)$, a straightforward way is to compute $m-1$ multiplications of $a_{ik} x_j^k, 1 \leq k \leq m-1$ by invoking the SS-multiplication protocol $m-1$ times. However, this will introduce too much communication cost.

Therefore, we propose to aggregate those multiplications into one round. That is, each party $P_l \in \mathcal{P}_i$ first locally compute a product-sum of shares $z_{ijl} = \sum_{k=1}^{m-1} [a_{ik}]_l [x_j^k]_l$ based on $m-1$ pairs of local shares $\{[a_{ik}]_l, [x_j^k]_l\}_{1 \leq k \leq m-1}$. Then, after computing z_{ijl}, each party $P_l \in \mathcal{P}_i$ proceeds in the same way as in SS-Mul. Specifically, each P_l shares the value z_{ijl} to others by choosing a t-degree random polynomial $h_l(x)$, and then locally computes the same linear combination $(\sum_{k=1}^{2t+1} \lambda_k h_k(l))$ of the received secondary shares to get its own share of the product-sum - $[\sum_{k=1}^{m-1} a_{ik} x_j^k]_l$. We denote this variant of SS-Mul as SS-Mul-Add, whose correctness follows from the homomorphic properties of SS-Add and SS-Mul. Since $F_i(x_j) = r_{ij} r'_{ij} (a_{i0} + \sum_{k=1}^{m-1} a_{ik} x_j^k + x_j^m) + x_j$, P_l's share of $F_i(x_j)$ can then be easily computed by invoking two more SS-Mul.

In the *reconstruction* phase, at least $t+1$ shares of $F_i(x_j)$ are needed to reconstruct $F_i(x_j)$. To this end, the parties reveal their shares to P_1 and P_i, who can obtain $F_i(x_j)$ by polynomial interpolation. P_1 and P_i can test if $F_i(x_j) = x_j, 1 \leq j \leq n$ and $F_i(x_j) = y_j, 1 \leq j \leq m$ respectively, to determine their intersection set.

C. The Advanced Scheme

Observe that, in the basic scheme if we set $F_i(x_j) = r_{ij} r'_{ij} f_i(x_j)$, then the result will be 0 if $x_j \in \mathcal{I}_{1,i}$, otherwise a random number. In order to obtain the number of matching attributes $(m_{1,i})$, one way is to employ the equality-test protocol [20] based on SS. However, this method incurs too high communication cost, since even the most efficient (probabilistic) algorithm takes $12k$ invocations of the SS multiplication protocol, where 2^{-k} is the error probability. When $k = 10$, the communicated bits for test one number $(F_i(x_j))$ amounts to $120N^2p$, and there are $(N-1)n$ numbers to test. Considering that in modern smart mobile devices, the wireless transmission is more costly than computation ability, we would like to tradeoff computation for communication efficiency.

Thus, we adopt a *blind-and-permute* (BP) method to obliviously permute P_1's shares of each $F_i(x_j)$, so that the linkage between $F_i(x_j)$ and its corresponding attribute x_j is broken. A BP protocol between two parties A and B where each data item (e.g., $F_i(x_j)$) is additively split between them is described in [21]. The main idea is, A encrypts each of its shares using additive homomorphic encryption and sends to B. B then generates a different random number r_j for each shared item, and randomizes each of A's shares by adding r_j, while subtracting r_j for its own corresponding shares. B permutes the randomized shares using a pseudo-random permutation (PRP), and sends back to A. All the computations are done over the ciphertexts.

However, the BP protocol cannot be applied directly. In our protocol, each $F_i(x_j)$ is polynomially shared among $t+1$ parties in $\mathcal{P}'_i$, where at most t of them may be adversarial. Without loss of generality, assume P_i is the party that generates the random permutation (π). Then in order to randomly

P_1's private inputs: $\mathcal{S}_1 = \{x_1, ..., x_n\}, \mathcal{I}_{1,i} \leftarrow \emptyset$; P_i's private inputs: $\mathcal{S}_i = \{y_{i1}, ..., y_{im}\}, \mathcal{I}'_{1,i} \leftarrow \emptyset$. Public inputs: $t, \mathcal{P}_i, \mathcal{P}'_i, 2 \leq i \leq N$. Output: $\mathcal{I}_{1,i}$ for both P_1 and P_i.

1. Data share distribution:

1) P_1 generates $\{r_{ij}\}_{1\leq j\leq n}$ randomly from $\mathcal{F}_p$. It shares $\{r_{ij}\}_{1\leq j\leq n}$ and $\{x_j, x_j^2, ..., x_j^m\}_{1\leq j\leq n}$ among parties in $\mathcal{P}_i$ using $(t, 2t+1)$-SS.
2) P_i generates $\{r'_{ij}\}_{1\leq j\leq n}$ randomly from $\mathcal{F}_p$, lets $f_i(y) = \prod_{k=1}^{m}(y - y_{ik}) = \sum_{k=0}^{m} a_{ik}y^k$, and shares $\{r'_{ij}\}_{1\leq j\leq n}$, $\{a_{ik}\}_{0\leq k\leq m-1}$ among parties in $\mathcal{P}_i$ using $(t, 2t+1)$-SS.

2. Computation: Each party $P_l \in \mathcal{P}_i \in \mathcal{P}$ executes the following:

1) $\forall 1 \leq j \leq n$, P_l computes the share of $\sum_{k=1}^{m-1}(a_{ik}x_j^k)$ through one invocation of SS-Mul-Add (namely, $[a_{i1}x_j + ... + a_{i,m-1}x_j^{m-1}]_l \leftarrow$ SS-Mul-Add$(\{[a_{ik}]_l, [x_j^k]_l\}_{1\leq k\leq m-1})$). (one communication round)
2) P_l computes $[f_i(x_j)]_l = [a_{i0}]_l + [a_{i1}x_j + ... + a_{i,m-1}x_j^{m-1}]_l + [x_j^m]_l$ via two SS-Add operations (local).
3) P_l computes $[R_{ij}]_l = [r_{ij}r'_{ij}]_l$ and $[F_i(x_j)]_l = [R_{ij}f_i(x_j) + x_j]_l$, for all $1 \leq j \leq n$, through two invocations of SS-Mul (two communication rounds), and one SS-Add (local).

3. Reconstruction:

1) Aggregate shares: each $P_l \in \mathcal{P}'_i$ sends its shares $\{[F_i(x_j)]_l\}_{1\leq j\leq n}$ to P_1 and P_i.
2) Reconstruction: P_1 reconstructs $F_i(x_j), 1 \leq j \leq n$ from their shares, using Lagrangian interpolation. If $F_i(x_j) \in \mathcal{S}_1$, $\mathcal{I}_{1,i} \leftarrow \mathcal{I}_{1,i} \bigcup F_i(x_j)$. The similar is done for P_i.

Fig. 1: The basic scheme, run between the initiator P_1 and each candidate $P_i \in \mathcal{P}, 2 \leq i \leq N$.

permute the shares of all the rest parties in $\mathcal{P}'_i$, there will be at least t invocations of the BP protocol between P_i and other parties in $\mathcal{P}'_i$, which is too computationally expensive.

Hence, in our scheme we propose an improvement that only requires *one* invocation of BP protocol. The idea is to convert the $(t, t+1)$-shares of $F_i(x_j)$ among $P_i \in \mathcal{P}'_i$ into $(1, 2)$-shares shared between P_1 and P_i. The conversion from $(t, t+1)$-share to $(1, t+1)$-share is a standard procedure [22] which involves one round of re-sharing. To transform $(t, t+1)$-shares into $(1, 2)$-shares, we use a trick in which all the parties in $\mathcal{P}'_i$ only re-share their $(t, t+1)$-shares of $F_i(x_j)$ to P_1 and P_i, so that *only P_1 and P_i together can reconstruct s*. Thus the BP protocol is performed only once between them. The parties P_1 and P_i then reconstruction the result $m_{1,i}$ by exchanging their shares. The protocol to compute $m_{1,i}$ is described in Fig. 2.

D. Personalizing Users' Privacy Levels

In our design, a user can choose her privacy level by telling other parties her choice in the beginning. For example, if a candidate P_i chooses PL-1, she can broadcast a message indicating "P_i selects PL-1". Then the parties in P_i's computing set and reconstruction sets will follow the basic scheme to compute the desired results for P_i. However, the initiator should always agree on the privacy level that each candidate proposes, since P_1 is at a position easier to conduct user profiling.

E. Practical Implementation Issues

In practice, proximity-based user discovery and key establishment are two important issues for the usability of our profile matching protocols. We envision that our FindU scheme can be used in mobile devices equipped with short-range wireless interfaces like WiFi or Bluetooth (most of today's smartphones have both interfaces), and operate in the ad-hoc mode. We have done some prior work in practical trust initialization in wireless networks [23]. Here we describe possible setup processes that involve little human effort.

User discovery If using Bluetooth, the existing Service Discovery Protocol (SDP) can be utilized to search for nearby FindU users (range is about 10m). As shown in [2], the SDP protocol can be used to publish/exchange information. We can use it to initiate the protocol (including key establishment). For WiFi, the ad-hoc mode would be sufficient for device discovery.

Key establishment As pairwise keys should be established between all nearby devices, a straightforward design (e.g., in Bluetooth each user needs to manually pair up with all the other users) would require $O(N^2)$ complexity. In order to reduce the complexity and minimize the need of explicit human participation, Diffie-Hellman key exchange (DHKE) can be used. Specifically, each device P_i chooses a random number $X_i \overset{R}{\leftarrow} \mathbb{Z}_p$ and publishes only $Y_i = g^{X_i}$ using broadcast. After receiving all the Y_j's, each P_i can calculate the pairwise key as P_j as $Y_j^{X_i}$. To achieve key authenticity, we can adopt tamper-evident pairing (TEP) [24], in which any modification of key exchange messages between two users by an attacker will be detected. Although TEP was implemented in the two-party setting, the same idea can also be applied to broadcast (each device in the group can negotiate to fix their messaging sequence and timing after device discovery and before pairing). In this method, the communication and time complexity is only $O(N)$. Though DHKE uses two modular exponentiation operations, this overhead can be amortized to multiple profile matching protocol runs. As in TEP, this approach works strictly in-band and only requires each user to press a button once, which is less intrusive. In addition, it is fully distributed and do not need any central server.

Indeed this approach is more favorable for WiFi devices with richer resources; however, for Bluetooth devices it can also be used. We note that although the SDP protocol does not support broadcast, each phone can publish up to 10 numbers with a maximum of 128 bytes each [2]. This would be sufficient for DHKE with 1024- bit group size. The downside is that unicast entails communication complexity of $O(N^2)$; however, given that the number of users within 10m range is

Inputs: basically the same as those in Fig. 1. In addition, P_1 sets $m_{1,i} = 0$, and has a public key pk_1 and private key sk_1, and pk_1 is known by all others. P_i sets $m'_{1,i} = 0$. Output: intersection set size $m_{1,i}$.
1. Data share distribution: the same as in Fig. 1, except that P_1 first randomly permutes its set S_1.
2. Computation: Basically the same as in Fig. 1, except each $P_l \in \mathcal{P}_i$ computes $[F_i(x_j)]_l = [R_{ij} f_i(x_j)]_l$, $\forall 1 \leq j \leq n$. For each $P_l \in \mathcal{P}'_i$, its shares are also denoted as $[F_i(x_j)]_l^{t,t+1}$.
3. Reconstruction:

1) Share conversion: Each $P_l \in \mathcal{P}'_i$ first computes and sends $[[F_i(x_j)]_l^{t,t+1}]_1^{1,t+1}$, $[[F_i(x_j)]_l^{t,t+1}]_i^{1,t+1}$ to P_1 and P_i, respectively. Then, P_1 and P_i compute $[F_i(x_j)]_1^{1,2}$ and $[F_i(x_j)]_i^{1,2}$, $1 \leq j \leq n$ using Lagrangian interpolation.
2) Blind and permute P_1's shares: P_1 and P_i involve in one BP-protocol where P_i generates permutation π and $\{r''_j\}_{1\leq j\leq n}$, and P_1's shares will be first randomized by $\{r''_j\}_{1\leq j\leq n}$ and then permuted by π.
3) Reconstruction: P_1 and P_i exchange their shares of $\{F_i(x_{j'})\}_{j'\in\{1,\cdots,n\}}$. Then, they reconstruct $F_i(x_{j'})$ for each j'. Then both count the number of $F_i(x_{j'}) = 0$, and set this number to $m_{1,i}$.

Fig. 2: The advanced scheme, run between P_1 and P_i.

usually small, it will not be a big problem.

V. Security Analysis

In this section, we first prove the security of each scheme under the HBC (semi-honest) model, and then discuss extensions to resist active attacks that deviate from the protocol run.

A. Security Definition

We define the security of our schemes based on the standard definition of secure multi-party computation under the HBC model [25] (c.f., Chapter 7), assuming private channels. Loosely speaking, "a multi-party protocol privately computes F, if whatever a set of semi-honest parties can obtain after participating in the protocol could be essentially obtained from the input and output of these very parties". Our protocols (denoted as Π) compute a two-ary functionality F (either the set intersection or cardinality of intersection set) between A (initiator) and B (a responder) with the help of at least $2t-1$ other parties (for simplicity, in our proofs we assume that all N parties $\mathcal{P}$ participate in their computation). That is, $F_(S_A, S_B) = S_A \bigcap S_B = \mathcal{I}_{A,B}$, and $F'_(S_A, S_B) = |S_A \bigcap S_B| = |\mathcal{I}_{A,B}|$ which are deterministic functions. Note that, the input of A is: $\vec{x}_A = S_A = \{x_1, ..., x_n\}$ while for B, the input is $\vec{x}_B = S_B = \{y_1, ..., y_m\}$. However, parties in $\mathcal{P}\backslash\{A, B\}$ do not have inputs nor yield any output. Thus, our protocol is a special case of multi-party computation. The view of the i-th party during an execution of Π on input $\bar{S} = \{S_A, S_B\}$ is defined as: $\text{VIEW}_i^{\Pi}(\bar{S}) \triangleq (S_i, r, m_1, ..., m_l)$ if $i = A$ or B, and $\text{VIEW}_i^{\Pi}(\bar{S}) \triangleq (r, m_1, ..., m_l)$ otherwise, where m_i represents the i-th message it has received and r stands for the internal randomness. Now we give formal definitions for privacy.

Definition 5 (t-privacy with respect to semi-honest behavior): We say that Π t-privately computes deterministic function F if there exist a P.P.T. algorithm, denoted as $\mathcal{A}$, such that for every coalition of semi-honest parties $\Gamma = \{i_1, ..., i_t\} \subset \mathcal{P}$ with size at most t, it holds that

$$\{\mathcal{A}(\Gamma, (\vec{x}_{i_1}, ..., \vec{x}_{i_t}), F_\Gamma(\vec{x}))\}_{\vec{x}\in\{0,1\}^*} \equiv \{\text{VIEW}_\Gamma^{\Pi}(\vec{x})\}_{\vec{x}\in\{0,1\}^*} \quad (2)$$

where $\vec{x}$ is the vector of all the inputs. In the middle when we write "$\equiv$", we mean perfect indistinguishability (correspondingly to computationally unbounded adversaries); while "$\overset{c}{\equiv}$" refers to computational indistinguishability (for computationally bounded adversaries).

In what follows, our proofs are based on simulation in the "real-world"/"ideal-world" paradigm.

B. Security Proof of the Basic Scheme Under the HBC Model

Theorem 1: The basic protocol (in Fig. 1) t-privately computes the set intersection between two parties A and B (in the semi-honest model), against computationally unbounded adversaries.

Proof: Here we give the proof sketch. The full proof is provided in Appendix. A.

The basic protocol Π_1 computes the set intersection ($F(S_A, S_B) = S_A \bigcap S_B$) between two parties A and B. It is realized by a multi-party protocol that first evaluates the function $F(x_j, S_B) = r_j r'_j f(x_j) + x_j$ on each of A's input $x_j, 1 \leq j \leq n$ and S_B, where $f(y) = \prod_{k=1}^{m}(y - y_k) = \sum_{k=0}^{m} a_k y^k$, and $\{r_j\}_{1\leq j\leq n}$, $\{r'_j\}_{1\leq j\leq n}$ are random blinding numbers chosen by A and B, respectively. Then A outputs $F_A(S_A, S_B) = S_A \bigcap \{F(x_j, S_B)\}_{1\leq j\leq n}$, and B outputs $F_B(S_A, S_B) = S_B \bigcap \{F(x_j, S_B)\}_{1\leq j\leq n}$.

The protocol Π_1 clearly consists of three phases: 1) input sharing; 2) share-based computation; 3) output reconstruction. All these phases are based on Shamir secret sharing. Our proof follows the idea in [26]. Intuitively, the protocol is t-private because the only values that the parties see until the output reconstruction phase are random shares. Since the threshold of the SS scheme is $t+1$, no adversary controlling up to t parties can learn anything. Due to the fact that t shares of any two secrets are identically distributed [26], the adversary's view can be simulated. In the third phase, the random numbers r_j and r'_j are both uniformly distributed and at least one of them remains secret to the adversary. Thus, except to match the inputs and final outputs of the adversary, the simulator can generate the shares based on any arbitrary value, while the resulting view is identical to that of a real execution.

The above intuition is formalized by constructing a simulator $\mathcal{A}$ for the view of an adversary that controls up to t parties Γ, which is only given the local inputs (and the security parameter 1^κ) and local outputs of the corresponding parties. There are three kinds of possible non-trivial situations: 1) $A \in \Gamma$, $B \notin \Gamma$; 2) $A \notin \Gamma$, $B \in \Gamma$; 3) $\{A, B\} \notin \Gamma$, related to different types of user profiling. Basically, to simulate the input sharing phase, the simulator first generates the inputs of honest parties randomly with the constraint of matching the corrupted parties' local outputs. Then it simulates the computation phase by emulating the real protocol run using those input shares. In both real/ideal worlds, the adversary's views consist of their own inputs and internal randomness, the messages received from honest parties, and (are augmented by) the intermediate shares they computed as the inputs/outputs to each wire of the arithmetic circuit in each round of computation, including the $\{F(x_j, S_B)\}_{1\le j\le n}$ values they obtained in the output reconstruction phase. Then, by induction method, we show that that the partial view (of any length) of parties in Γ in a real execution is identically distributed to the partial view (of the same length) output by the simulator, and also the full views up to the output reconstruction phase. Note that for an adversary that does not control neither A nor B, i.e., case 3), since it does not receive any inputs/outputs, everything it sees during either real/ideal executions is less than or equal to t random shares. Thus the perfect indistinguishability follows in all the cases. ■

Remark on security parameters. Due to unconditional security, it is enough as long as the field $\mathcal{F}_p$ can represent all the attributes in the attribute dictionary. Therefore, assume there are 1×10^6 attributes, we choose $\Bbbk = \log p = 24$.

C. Security Proof of the Advanced Scheme Under the HBC Model

Theorem 2: Assuming a secure pseudorandom permutation and semantically secure additive homomorphic encryption scheme, the advanced protocol (in Fig. 2) t-privately computes the cardinality of set intersection between two parties A and B (in the semi-honest model), against computationally bounded adversaries.

Proof: Proof sketch. We again construct a simulator $\mathcal{A}$ that simulates the adversary Γ's view, and we show that the real/ideal views are computationally indistinguishable. We still use the induction method as in the previous proof. Essentially the *input sharing* and *share-based computation* phases remain the same as in the basic protocol, so we only need to focus on the output reconstruction phase. In the *share conversion* round, the adversary's appended real/ideal views consist of only one share of values that are chosen from a random 1-degree polynomial, which are in fact identically distributed conditioned on real/simulated partial views of the previous rounds. In the following, the real executions are only seen by A and B. In the *blind-and-permute* round, if $A \in \Gamma, B \notin \Gamma$, A's received randomized and permuted shares are also uniformly distributed and independent from the previous partial views. If $B \in \Gamma, A \notin \Gamma$, due to the semantic security of the additive homomorphic encryption scheme, distributions of the received encrypted messages by B are computationally indistinguishable. Finally, the partial views of Γ in the reconstruction phase are computationally indistinguishable in both cases (given the previous partial views), because the the distributions of the reconstructed values $\{F'(x_j, S_B) = r_j r'_j f(x_j)\}_{1\le j\le n}$ (including $|\mathcal{I}_{A,B}|$ zeros and $n - |\mathcal{I}_{A,B}|$ uniformly distributed random numbers) are indistinguishable due to the security of the pseudorandom permutations. The full proof can be found in Appendix. B. ■

D. Discussion: Preventing Malicious Attacks

Our protocols in this paper are only proven secure in the HBC model; it would be interesting to make it secure under the stronger malicious model, i.e., to prevent an adversary from arbitrarily deviating from a protocol run. In the conference version of this paper [1], we showed that with an additional commitment round before final reconstruction (which adds little additional overhead), a specific type of "set inflation attack" can be easily prevented where a malicious user influences the final output in her favorable way by changing her shares after seeing others'.

Further, we observe it is possible that our protocols can be extended to the malicious model with some additional cost[1], following the same method in [26]. The idea is to use verifiable secret sharing (VSS) [18], [19], which is secure under the malicious model with $t < N/3$ malicious parties. Asharov and Lindell proved in [26] that (c.f. Sec. 7), with a VSS scheme and a secure multiplication protocol F_{mul} that are both secure under the malicious model, an arbitrary multi-party function F can be computed securely in the malicious model. In addition they proposed a construction of F_{mul} that is based on VSS. They represent F using an arithmetic circuit with three kinds of gates: addition, multiplication-by-constant, and multiplication; secure computation of F replaces each of those gate by its secret sharing counterpart, namely SS-Add, SS-Mul-by-Const, SS-Mul. As long as these components are secure, they can be composed into a secure protocol according to the composition theorem [27].

Therefore, in our situation, as long as we design a secure SS-based Mul-Add protocol that is secure in the malicious model, our PSI and PCSI protocols can be extended to be secure under the malicious model using VSS. Because the Mul-Add aggregates multiple multiplications into one round, we believe the same method to construct a secure F_{mul} still applies. This will be left as an interesting future work.

VI. Performance Evaluation

In this section, we analyze the complexity of our proposed schemes[2], carry out an extensive simulation study of the protocols' efficiency, and compare them with several state-of-the-art schemes in terms of security and efficiency.

A. Complexity Analysis

1) Computational cost: The computational cost is evaluated using the number of modular multiplication and exponentiation operations, while the communication cost is calculated in terms of number of transmitted/received bits. Let $\Bbbk = 24$, and

[1]Theoretically, a general SMC protocol that is secure in the semi-honest model can be converted to one secure in the malicious model, using a "compiler" [25], which, however, bears too much cost.

[2]To reflect the reality, the full matching protocols between the initiator and $N-1$ other users are considered, which are aggregations of $N-1$ individual protocol runs between P_1 and each P_i [1].

TABLE II: Comparison of complexity ($q = 1024$, $\Bbbk = 24$)

	Party	Basic scheme	PSI [7]	Advanced scheme	PCSI [5]
Computation	P_1	$6nNt + mnN(1 + t\log N/\Bbbk)$ mul_1	$1.5nN$ mul_2	$3nN$ exp_3	$(2n + mN)$ exp_3
	P_i	$2mnt + 2(m + 6nt)t^2\log N/\Bbbk$ mul_1	$(m + n)$ exp_2	n exp_3	$m\log(\log n)$ exp_3
Comm. (TX)	P_1	$(mnN + 8nNt)\Bbbk$	$2nNq$	$(mnN + 8nNt)\Bbbk + 2nNq$	$2nq$
	P_i	$2t(m + 6nt)\Bbbk$	$(n + m)q$	$2t(m + 6nt)\Bbbk + 2nq$	$2mq$
Comm. (RX)	P_1	$[N(m + n) + 8nNt]\Bbbk$	$(n + m)Nq$	$N[m + (t + 3N)n]\Bbbk + 2nNq$	$2mNq$
	P_i	$[m(n + 2t) + 12nt^2]\Bbbk$	$2nq$	$[m(n + 2t) + 12nt^2]\Bbbk + 2nq$	$2nq$
Comm. total	All	$[mnN + tN(8n + 2m + 12nt)]\Bbbk$	$N(3n + m)q$	$[mnN + tN(8n + 2m + 12nt)]\Bbbk + 4nNq$	$2(n + mN)q$

assume one $\Bbbk$-bit modular multiplication (denoted as mul_1) takes $\Bbbk^2$ bit operations. Thus, to share a secret $s \in \mathcal{F}_p$ among N parties using (t, N)-SS, it takes $\Bbbk Nt\log N$ bit operations [22], which is $\frac{tN\log N}{\Bbbk}$ multiplications. For 1024-bit and 2048-bit modular multiplications (denoted as mul_2, mul_3) and exponentiations (denoted as exp_2, exp_3), since there exist optimization algorithms, we will use existing benchmark test results instead. Note that we neglect modular additions, and we make the following assumptions to simplify the calculations: $m \gg N$, $q \gg \Bbbk$ and $m, n, N, t \gg 1$. For the details of the complexity analysis, please refer to our technical report [28].

Recall that the computing set and the reconstruction set for each $P_i, 2 \leq l \leq N$ have sizes $2t + 1$ and $t + 1$, respectively. Also, each $P_i, 2 \leq l \leq N$ is in $2t$ parties' computing sets and is in t parties' reconstruction sets. In phase 1 (data share distribution), P_1 distributes $2tn + mn$ numbers to N party, which takes $(4t^2n + mnN)t\log N/\Bbbk$ mul_1 operations. For $P_i, 2 \leq l \leq N$, this is $2(n + m)t^2\log N/\Bbbk$ instead. In phase 2, in each step, P_1 takes $(m - 1)nN$, $2nNt^2\log N/\Bbbk$, $2nNt$ and $2nN[1 + 2t(1 + t\log N/\Bbbk)]$ mul_1 operations. For P_i, the numbers are $2(m - 1)nt$, $4nt^3\log N/\Bbbk$, $4nt^2$ and $4nt[1+2t(1+t\log N/\Bbbk)]$. In phase 3, P_1 takes $n(N-1)(t+1)$ mul_1, while P_i takes $n(t + 1)$. Therefore, the total computational cost for P_1 is about $6nNt + mnN(1 + t\log N/\Bbbk)$ mul_1, and $2mnt + 2(m + 6nt)t^2\log N/\Bbbk$ mul_1 for P_i.

For the advanced scheme, the costs up to the computation phase remains the same as in the basic scheme. In the reconstruction phase, the major computation cost of each party comes from homomorphic encryption and decryption. A public key encryption scheme must be used; for example, the Paillier's cryptosystem uses 2048-bit modulus and thus requires two 2048-bit exponentiations for encryption and one for decryption. We denote the length of ciphertext as $2q, q = 1024$ (in order to compare with other schemes). P_1 incurs $2nN$ exp_3 operations for encrypting nN numbers in step 3.2, and nN for decryption. For P_i, its computation cost is dominated by n encryptions, i.e., $2n$ exp_3.

2) Communication cost: The communication cost is evaluated in terms of number of transmitted bits. For the basic scheme, in phase 1, P_1 transmits $nN(2t + m)\Bbbk$ bits, while P_i transmits $2t(m + n)\Bbbk$ bits. In phase 2, P_1 sends $6nNt\Bbbk$ bits, while P_i sends $12nt^2\Bbbk$ bits. In phase 3, the communication costs are negligible compared with previous phases. Thus, in total P_1 transmits approximately $(mnN + 8nNt)\Bbbk$ bits, and P_i transmits $2t(m + 6nt)\Bbbk$. For all parties, their sum of communication cost is $[mnN + tN(8n + 2m + 12nt)]\Bbbk$.

For reception, P_1 receives $[N(m+n)+8nNt]\Bbbk$ bits in total, while P_i receives about $[m(n + 2t) + 12nt^2]\Bbbk$ bits.

For the advanced scheme, the costs up to the computation phase are the same as in the basic scheme. The communication cost of P_1 in the reconstruction phase is $nN(3\Bbbk + 2q)$, where $2q$ is the ciphertext length of homomorphic encryption. For P_i, the cost is $2nt\Bbbk + n(2\Bbbk + 2q)$.

The complexity results are summarized in Table II, and we compare our basic and advanced schemes with existing schemes: FC10 PSI [7] and FNP PCSI (under HBC model) [5], respectively. It can be seen that, the basic scheme's total computation complexity is much smaller than that of FC10's since $q \gg \Bbbk$, while that of the advanced scheme's is smaller than FNP's when n is relatively small w.r.t. m. Although the total communication costs may seem large in our schemes, they are on the same orders with the compared schemes in terms of m, n and N. The effect of the $O(t^2)$ factor is moderate unless t scales linearly with N, as we will see in the simulation results.

B. Simulation Study

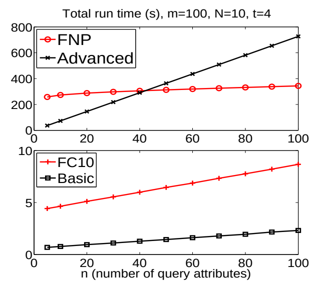
1) Methodology: We implement our proposed schemes and two previous schemes, FC10 and FNP, in NS-2 [29]. We simulate the protocols' communications and computations by telling the simulator the sizes and number of packets each party should send, fill each packet with dummy contents, and estimate the latency of each computation. Note that, in each round/step, we exploit the opportunities to aggregate messages sent to the same party into one packet as much as possible, so as to reduce the number of packets sent. In addition, we only simulate one full protocol run, as the time variance is very small due to deterministic scheduling.

For simulation settings, we assume the use of WiFi as the wireless interface which operates in the ad-hoc mode, and IEEE 802.11a is used for MAC and PHY layer. Nodes (parties) are randomly distributed in a $50m \times 50m$ area. The transmission range is set to 200m, such that all nodes are within reach of each other. Two-ray-ground propagation model is assumed, and the wireless channels are reliable.

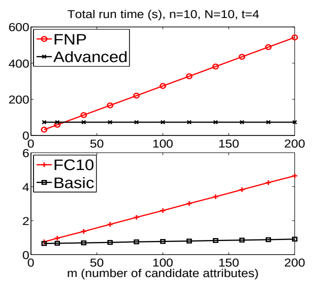
Evaluation metrics. To evaluate the efficiency of the protocols, we adopt energy consumption and total run time as two unified metrics. Both of them factor into the effects of both computation and communication time, and are closely relate to the user experience; the energy consumption is also affected by the total run time.

We estimate the computation time for primitive operations based on the existing benchmark test results [30]. Assuming a 400MHz CPU, the times for the primitive crypto operations are (seconds): 1.4×10^{-6} for mul_1, 8×10^{-5} for mul_2, 0.04 for exp_2, 2.4×10^{-4} for mul_3, and 0.25 for exp_3.

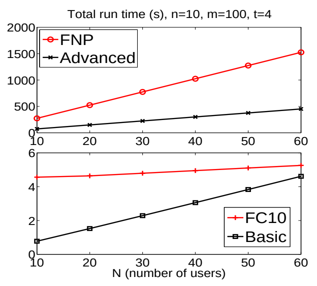
Our energy consumption model is based on [31]: $E =$



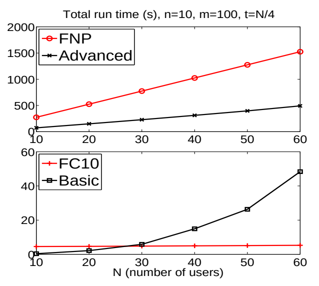
(a) Change n.



(b) Change m.

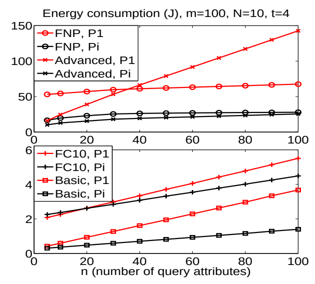


(c) Change N, fix $t = 4$.

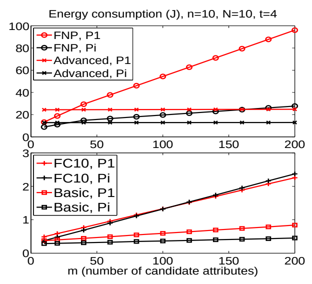


(d) Change N, change $t = \lfloor N/4 \rfloor$.

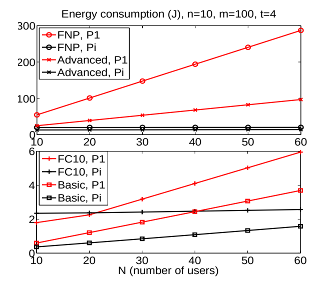
Fig. 3: Total protocol run time. (Y-axis: protocol run time (s).)



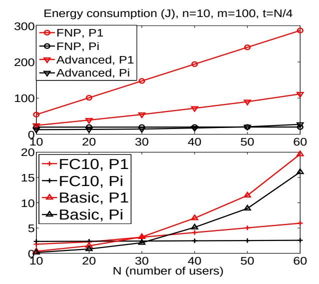
(a) Change n.



(b) Change m.



(c) Change N, fix $t = 4$.



(d) Change N, change $t = \lfloor N/4 \rfloor$.

Fig. 4: Energy consumption for P_1 and P_i. (Y-axis: energy consumption (J).)

$$E = \begin{cases} 10^{-6}(6.7n_t + 4.8n_r) + P_{comp}T_{comp} + 0.3167T_{total} \quad (J), & \text{If computations in every step take time less than } 15s \\ 10^{-6}(6.7n_t + 4.8n_r) + P_{comp}T_{comp} + 0.3167(T_{total} - T_{shut}) + 5n_{shut} \quad (J), & \text{Otherwise.} \end{cases} \tag{3}$$

$n_tE_t + n_rE_r$ [3], where n_t and n_r are sent and received in bytes, $E_r \approx 6.7\mu J$ is the receiving energy per byte, and $E_t \approx 4.8\mu J$ is transmitting energy per byte. Also, if the connection time exceeds 15 seconds, it is more efficient to shut down WIFI radio [32]. Thus, in simulation we employ the following strategy to save energy: for each party, whenever it estimates its computing time in one step to be longer than 15s, it shuts down its own WIFI radio and also indicates others to close theirs for a known time period, and reopens it when the computation is done. Now, the model can be described in Eq. (2), where n_{shut} is the # of times WIFI radio has shut down, T_{shut} is the total radio shut down time, P_{comp}, T_{comp} are the CPU's power consumption and time spent for computation, and T_{total} stands for the total protocol run time. For a smart phone with 400MHz CPU, we choose $P_{comp} \approx 0.18$w [33].

2) *Simulation results:* We show the simulation results in Fig. 3 and Fig. 4; we compare the basic scheme with the FC10 scheme [7] (the RSA exponents/modulus are both 1024-bits), and the advanced scheme with the FNP scheme [5]. We also assume the use of Paillier's crytosystem for FNP, with 2048 bit modulus. Note that, we use $\Bbbk = 24$ bits for our schemes.

In Fig. 3 (a) and Fig. 4 (a), we fix the network size $N = 10$, maximum number of colluders $t = 4$, number of profile attributes $m = 100$, and change number of query attributes n. It can be seen that, the basic scheme takes less than 1 second when $n < 100$. Both the total run time and energy consumption of it is much less than that of FC10's in this case and they all increase linearly with n. For the advanced scheme, the time and energy are smaller than those of FNP's when $n < 40$. This is mainly because the computation in the advanced scheme scales as $O(nN)$, which is $O(n + mN)$ for the FNP scheme. Nevertheless, for the profile matching application, in reality it is often the case that n is small.

On the other hand, from Fig. 3 (b) and Fig. 4 (b) in which n is fixed to 10 but m changes from 10 to 200, it can be see that both the basic and the advanced schemes are more efficient than their counterparts, more importantly our proposed schemes are hardly affected by m. The above shows that our schemes are more practical when the number of profile attributes is large, while the number of query attributes is small.

Next, we change the number of parties. We first fix the maximum number of colluders to 4. From Fig. 3 (c) and Fig. 4 (c), one can observe that the basic scheme's costs increase linearly with N. This is because its run time and energy costs are both dominated by communication which is linear in N when t is fixed, since the computations are quite fast. The FC10 scheme is much more computationally intensive under this case. For the advanced scheme and FNP, their costs are both dominated by computation rather than communication, yet the advanced scheme performs much better due to $n < m$.

[3]We omit the initial connection establishment energy since it is common in all schemes.

TABLE III: Comparison of security, and computation/communication efficiency under typical parameters (n=10, m=100, N=10)

Schemes		Basic	Advanced	PSI [7]	PSI [34]	PCSI [5]	PSI [11]	PCSI [11]
Adversary Model		HBC	HBC	HBC	Malicious	HBC	HBC	HBC
Resist active attacks		No*	No*	No	Yes	No	No	No
Computation	P_1	1.6×10^4 mul_1	300 exp_3	250 mul_2	$4 \times 10^4 \mathsf{mul}_2$	1020 exp_3	6×10^4 mul_1	4.4×10^4 mul_1
	P_i	9.5×10^3 mul_1	20 exp_3	110 exp_2	$2.96 \times 10^4 \mathsf{mul}_2$	173 exp_3	4.5×10^4 mul_1	3.5×10^4 mul_1
Communication (KB) Sent/Received	P_1	$40/13$	$76/39$	$26/141$	$37/127$	$2.6/256$	$280/253$	$528/528$
	P_i	$8.2/11$	$11/13.6$	$14/2.6$	$14/4.1$	$25.6/2.6$	$200/203$	$422/422$
Total comp. time (s)		0.023	80	4.42	5.57	298	0.14	0.093
Total sent bytes (KB)		114	175	166	164	259	2080	4326
Note: No* means can be easily extended to prevent certain malicious attacks.								

P_i has almost constant energy consumption except in the basic scheme, since in those three schemes P_i's computation cost is mainly affected by m, but not N.

Finally, we scale t with N ($t = \lfloor N/4 \rfloor$) in Fig. 3 (d) and Fig. 4 (d). Interestingly, the advanced scheme is still much more efficient than the FNP scheme, and it exhibits a super-linear effect ($O(t^3)$ in overall communication) only when $N > 50$ for P_i's energy consumption. Meanwhile, the basic scheme suffers from this effect earlier than the advanced scheme (better than FC10 when $N < 30$), since it is dominated by communication.

The above demonstrates the advantage of our proposed schemes when n and N are both relatively small (in the order of tens), which is usual in mobile social networks. Note that, in all the compared protocols, it is always the case that P_1 uses more energy than P_i. Through using the energy-saving strategy, for the advanced scheme the parties' energy consumption seldom exceed 100J (equivalent to purely using WIFI for 5min), while that of the basic scheme is below 10J.

C. Further Comparison with Previous Works

We now compare our schemes with more existing schemes, in terms of security and efficiency. The computation/communication complexities are calculated by aggregating $N-1$ protocol runs between P_1 and each P_i, with P_1 being the initiator/client and each P_i being a responder/server.

First, the PSI scheme in [34] achieves standard security under the malicious adversary model. Its computation complexity is $400n(N-1)\mathsf{mul}_2$ operations for P_1 and $(560n+240m)\mathsf{mul}_2$ for P_i. For sent bytes, those are $(3n+2)(N-1)q$ for P_1 and $(2n+m+1)q$ for P_i. While they are linear with n, m, the constant factors are much higher than our basic scheme under typical MSN scenarios.

The schemes in [11] (CANS'09) represent a category with unconditional security. Their PSI and PCSI schemes are also based on secret sharing. For the PSI scheme, we modify it to fit our problem setting (by adding r_{ij} and r'_{ij}), which differs from our basic scheme in that: (1) it does not aggregate the multiplications in the computation phase; (2) it does not prevent set inflation attack. We denote the computation complexity of P_1 in our basic scheme as $\text{Comp}_{P_1}(\text{Basic})$, the number of bits sent by P_1 in our basic scheme as $\text{Comm}_{P_1}(\text{Basic})$, while $\text{Comm}'_{P_1}(\text{Basic})$ stands for the number of bits received by P_1 in our basic scheme. The computation complexity for the PSI scheme in [11] is $\text{Comp}_{P_1}(\text{Basic}) + 2mnNt^2 \log N / \Bbbk$ mul_1 operations for P_1, and $\text{Comp}_{P_i}(\text{Basic}) + 4mnt^3 \log N / \Bbbk$ mul_1 for P_i. The bytes sent by P_1 and P_i are: $\text{Comm}_{P_1}(\text{Basic}) + 2mnNt\Bbbk$, and $\text{Comm}'_{P_1}(\text{Basic}) + 4mnt^2\Bbbk$, respectively. For the PCSI scheme in [11], computation complexity for P_1 and P_i are: $2mnNt^2 \log N / \Bbbk$ mul_1 and $4mnt^3 \log N / \Bbbk$ mul_1, respectively, while the bytes sent are $(2mnNt + 240nNt)\Bbbk$ and $(4mnt^2 + 480nt^2)\Bbbk$, respectively.

The security and efficiency comparisons are summarized in Table III. For efficiency comparison, we set the parameters to be typical values: $n = 10$, $m = 100$, $N = 10$ and $t = 4$, and numerically evaluate the computation and communication costs. Note that, we also count the offline computations since those also consume energy. From Table III, it can be seen that our basic scheme outperforms all the other PSI schemes, while the advanced scheme is slower in computationally than the PCSI scheme in [11]. However, the latter is expensive in communication, and does not protect against active attacks.

VII. Conclusion

In this paper, we for the first time formalize the problem of privacy-preserving distributed profile matching in MSNs, and propose two concrete schemes that achieve increasing levels of user privacy preservation. Towards designing lightweight protocols, we utilize Shamir secret sharing as the main secure computation technique, while we propose additional enhancements to lower the proposed schemes' communication costs. Through extensive security analysis and simulation study, we show that 1) our schemes are proven secure under the HBC model, and can be easily extended to prevent certain active attacks; 2) our schemes are much more efficient than state-of-the-art ones in MSNs where the network size is in the order of tens, and when the number of query attributes is smaller than the number of profile attributes.

Acknowledgment

This work was supported in part by the US National Science Foundation under grants CNS-0716306, CNS-0831628, and CNS-0746977. The authors would like to thank Prof. William Martin for helpful discussions at the early stage of the protocols.

References

[1] M. Li, N. Cao, S. Yu, and W. Lou, "Findu: Privacy-preserving personal profile matching in mobile social networks," in *IEEE INFOCOM '11*, Apr 2011, pp. 1–9.

[2] 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 '10*, June. 2010.

[3] 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.

[4] C. Zhang, X. Zhu, Y. Song, and Y. Fang, "C4: A new paradigm for providing incentives in multi-hop wireless networks," in *INFOCOM, 2011 Proceedings IEEE*, april 2011, pp. 918 –926.

[5] M. Freedman, K. Nissim, and B. Pinkas, "Efficient private matching and set intersection," in *EUROCRYPT'04*. Springer-Verlag, 2004, pp. 1–19.

[6] Q. Ye, H. Wang, and J. Pieprzyk, "Distributed private matching and set operations," in *ISPEC'08*, 2008, pp. 347–360.

[7] E. D. Cristofaro and G. Tsudik, "Practical private set intersection protocols with linear complexity," in *Financial Cryptography and Data Security '10*, 2010.

[8] L. Kissner and D. Song, "Privacy-preserving set operations," in *CRYPTO '05, LNCS*. Springer, 2005, pp. 241–257.

[9] A. C. Yao, "Protocols for secure computations," in *SFCS '82*, 1982, pp. 160–164.

[10] D. Dachman-Soled, T. Malkin, M. Raykova, and M. Yung, "Efficient robust private set intersection," in *ACNS '09*, 2009, pp. 125–142.

[11] 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 '09*. Springer - Verlag, Dec. 2009, pp. 21–40.

[12] C. Hazay and Y. Lindell, "Efficient protocols for set intersection and pattern matching with security against malicious and covert adversaries," in *TCC'08*, 2008, pp. 155–175.

[13] S. Jarecki and X. Liu, "Efficient oblivious pseudorandom function with applications to adaptive ot and secure computation of set intersection," in *TCC '09*. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 577–594.

[14] W. Dong, V. Dave, L. Qiu, and Y. Zhang, "Secure friend discovery in mobile social networks," in *IEEE INFOCOM '11*, Apr 2011, pp. 1–9.

[15] E. De Cristofaro, M. Manulis, and B. Poettering, "Private discovery of common social contacts," in *Applied Cryptography and Network Security*. Springer, 2011, pp. 147–165.

[16] E. De Cristofaro, A. Durussel, and I. Aad, "Reclaiming privacy for smartphone applications," in *Pervasive Computing and Communications (PerCom), 2011 IEEE International Conference on*, march 2011, pp. 84 –92.

[17] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[18] M. Ben-Or, S. Goldwasser, and A. Wigderson, "Completeness theorems for non-cryptographic fault-tolerant distributed computation."

[19] R. Gennaro, M. O. Rabin, and T. Rabin, "Simplified vss and fast-track multiparty computations with applications to threshold cryptography," in *ACM PODC '98*, 1998, pp. 101–111.

[20] T. Nishide and K. Ohta, "Multiparty computation for interval, equality, and comparison without bit-decomposition protocol," in *PKC'07*, 2007, pp. 343–360.

[21] Y. Qi and M. J. Atallah, "Efficient privacy-preserving k-nearest neighbor search," in *IEEE ICDCS '08*, 2008, pp. 311–319.

[22] E. Kiltz, "Unconditionally secure constant round multi-party computation for equality, comparison, bits and exponentiation," in *TCC '05*. Springer, 2005.

[23] M. Li, S. Yu, J. D. Guttman, W. Lou, and K. Ren, "Secure ad-hoc trust initialization and key management in wireless body area networks," *ACM Transactions on Sensor Networks (TOSN)*, 2012.

[24] S. Gollakota, N. Ahmed, N. Zeldovich, and D. Katabi, "Secure in-band wireless pairing," in *Proceedings of the 20th USENIX conference on Security*, ser. SEC'11, 2011, pp. 16–16.

[25] O. Goldreich, *Foundations of Cryptography: Volume 2, Basic Applications*. Cambridge Univ Pr, 2009.

[26] G. Asharov and Y. Lindell, "A full proof of the bgw protocol for perfectly-secure multiparty computation," *Advances in CryptologyCRYPTO 2011*, 2011.

[27] R. Canetti, "Security and composition of multiparty cryptographic protocols," *Journal of Cryptology*, vol. 13, pp. 143–202.

[28] M. Li, S. Yu, N. Cao, and W. Lou, "Privacy-preserving distributed profile matching in proximity-based mobile social networks," technical Report, 2012, http://digital.cs.usu.edu/~mingli/papers/findu_techrep.pdf.

[29] "Ns2," *http://www.isi.edu/nsnam/ns*.

[30] S. Bhatt, R. Sion, and B. Carbunar, "A personal mobile drm manager for smartphones," *Computers & Security*, vol. 28, no. 6, pp. 327 – 340, 2009.

[31] A. Rahmati and L. Zhong, "Context-for-wireless: context-sensitive energy-efficient wireless data transfer," in *MobiSys '07*, 2007, pp. 165–178.

[32] R. Balani, "Energy consumption analysis for bluetooth, wifi and cellular networks," in *Technical report*, Dec. 2007, pp. 1–6.

[33] A. Carroll and G. Heiser, "An analysis of power consumption in a smartphone," in *Usenix technical conference*, Boston, MA, USA, Jun 2010.

[34] G. T. E. De Cristofaro, J. Kim, "Linear-complexity private set intersection protocols secure in malicious model." in *Asiacrypt*, 2010.

[35] R. Cramer, I. Damgard, and J. Nielsen, "Secure multiparty computation," *Book draft*, 2010.

Appendix A
Detailed Proof of Security of our Basic Protocol Under the HBC Model

Theorem 3: The basic protocol (in Fig. 1) t-privately computes the set intersection between two parties A and B (in the semi-honest model), against computationally unbounded adversaries.

Proof: The basic protocol Π_1 computes the set intersection ($F(S_A, S_B) = S_A \bigcap S_B = F_A(S_A, S_B) = F_B(S_A, S_B)$) between two parties A and B. It is realized by a multiparty protocol that first evaluates the function $F(x_j, S_B) = r_j r'_j f(x_j) + x_j$ on each of A's input $x_j, 1 \le j \le n$ and S_B, where $f(y) = \prod_{k=1}^{m}(y - y_k) = \sum_{k=0}^{m} a_k y^k$, and $\{r_j\}_{1\le j\le n}$, $\{r'_j\}_{1\le j\le n}$ are random blinding numbers chosen by A and B, respectively. Then A outputs $F_A(S_A, S_B) = S_A \bigcap \{F(x_j, S_B)\}_{1\le j\le n}$, and B outputs $F_B(S_A, S_B) = S_B \bigcap \{F(x_j, S_B)\}_{1\le j\le n}$.

The protocol Π_1 clearly consists of three phases: 1) input sharing; 2) share-based computation; 3) output reconstruction. All these phases are based on Shamir secret sharing. Our proof follows the idea in [26], [35]. Intuitively, the protocol is t-private because the only values that the parties see until the output reconstruction phase are random shares. Since the threshold of the SS scheme is $t + 1$, no adversary controlling t parties can learn anything. Due to the fact that t shares of any two secrets are identically distributed [26], the adversary's view can be simulated. In the third phase, the random numbers r_j and r'_j are both uniformly distributed and at least one of them remains secret to the adversary. Thus, except to match the inputs and final outputs of the adversary, the simulator can generate the shares based on any arbitrary value, while the resulting view is identical to that of a real execution.

Next we formalize our intuition by constructing a simulator $\mathcal{A}$ for the view of an adversary that controls up to t parties Γ, which is only given the local inputs (and the security parameter

1^κ) and local outputs of the corresponding parties. There are three kinds of possible non-trivial situations: 1) $A \in \Gamma$, $B \notin \Gamma$; 2) $A \notin \Gamma$, $B \in \Gamma$; 3) $\{A, B\} \notin \Gamma$, related to different types of user profiling. We focus on describing a simulator for case 1), while other cases follow easily from it. The simulator proceeds as follows:

- **Input**: The simulator receives inputs and outputs, ($S_A = \{x_1, ..., x_n\}, 1^\kappa$) and $\mathcal{I}_{A,B}$, respectively, from $A \in \Gamma$ (the other parties' inputs/outputs are empty).
- **Simulation**:
  - Simulating the *input sharing* phase:
    (1) For party A, $\mathcal{A}$ first computes $\{x_j, x_j^2, ..., x_j^m\}_{1\le j\le n}$ from input, and then chooses a uniformly distributed random tape, from which it generates n random numbers $\{r_j\}_{1\le j\le n} \overset{R}{\leftarrow} \mathcal{F}_p$, and also determines the degree-t polynomials $\{p'_{j,k}(x) \in \mathcal{P}^{x_j^k,t}\}_{1\le k\le m, 1\le j\le n}$, $\{p''_j(x) \in \mathcal{P}^{r_j,t}\}_{1\le j\le n}$ chosen by A, where $\mathcal{P}^{x,t}$ is the set of all polynomials with degree less than or equal to t with free coefficient x.
    (2) For party B, it chooses $m - |\mathcal{I}_{A,B}|$ random numbers $y'_1, ..., y'_{m-|\mathcal{I}_{A,B}}$ from $\mathcal{F}_p$ as a set S'_B. Then it sets $S'_B = S'_B \bigcup \mathcal{I}_{A,B} = (y'_1, ..., y'_m)$. It also selects random numbers $\{\tilde{r}'_j\}_{1\le j\le n} \overset{R}{\leftarrow} \mathcal{F}_p$, and sets $\{a'_k\}_{0\le k\le m-1}$ to be the coefficient of $f'(y) = \prod_{k=1}^m (y - y'_k)$ ($a'_m = 1$). Now it generates random polynomials $q'_k(x) \in \mathcal{P}^{a'_k,t} (0 \le k \le m-1)$, and $q''_j(x) \in \mathcal{P}^{\tilde{r}'_j,t} (1 \le j \le n)$.
    (3) The view of parties in Γ is recorded to be $\{\{p'_{j,k}(i), p''_j(i), q''_j(i)\}_{1\le k\le m, 1\le j\le n}, \{q'_k(i)\}_{0\le k\le m-1}, \}_{P_i\in\Gamma}$ and the random tape chosen for $A \in \Gamma$.
  - Simulating the *share-based computation* phase: For every round of computation[4]:
    (1) In the SS-Mul-Add round (in the real world each party executes n product-sum computation in parallel, where each can be viewed as an arithmetic circuit of $m-1$ multiplication gates and $m-2$ addition gates), let the input shares of each $P_i \in \mathcal{P}$ on the j-th circuit be $(p'_{j,1}(i), q'_1(i), ..., p'_{j,m-1}(i), q'_{m-1}(i))$. For each $P_i \in \mathcal{P}$, $\mathcal{A}$ computes product-sums $\{z_{ij} = \sum_{k=1}^{m-1} p'_{j,k}(i) q'_k(i)\}_{1\le j\le n}$, and chooses n random polynomials $\{h_{i,j}(x) \in \mathcal{P}^{z_{ij},t}\}_{1\le j\le n}$. The view of $P_i \in \Gamma$ is appended by $\{h_{l,j}(i)\}_{P_l\in\mathcal{P}, 1\le j\le n}$ and $\{H_j(i) = \sum_{P_l\in\mathcal{P}} \lambda_l h_{l,j}(i) = (\sum_{P_l\in\mathcal{P}} \lambda_l h_{l,j})(i)\}_{1\le j\le n}$ (output shares)[5].
    (2) In the following two SS-Add rounds, the input shares of each $P_i \in \Gamma$ are $\{H_j(i), p'_{j,m}(i), q'_0(i)\}_{1\le j\le n}$. For every $P_i \in \Gamma$, $\mathcal{A}$ simply adds them together, and record their output shares as $\{\tilde{g}_j(i) = (H_j + p'_{j,m} + q'_0)(i)\}_{1\le j\le n}$ (i.e., the shares of $\{f'(x_j)\}_{1\le j\le n}$), and append to view of $P_i \in \Gamma$.
    (3) In the following SS-Mul round, the input shares of each $P_i \in \mathcal{P}$ are $\{p''_j(i), \tilde{g}_j(i)\}_{1\le j\le n}$. For each $P_i \in \mathcal{P}$, $\mathcal{A}$ computes the products $\{p''_j(i)\tilde{g}_j(i)\}_{1\le j\le n}$, and chooses n random polynomials $\{\tilde{h}'_{i,j}(x) \in \mathcal{P}^{p''_j(i)\tilde{g}_j(i),t}\}_{1\le j\le n}$. The view of $P_i \in \Gamma$ is appended by $\{\tilde{h}'_{l,j}(i)\}_{P_l\in\mathcal{P}, 1\le j\le n}$ and $\{\tilde{H}'_j(i) = (\sum_{P_l\in\mathcal{P}} \lambda_l \tilde{h}'_{l,j})(i)\}_{1\le j\le n}$. Similarly, for the other SS-Mul round, the view of $P_i \in \Gamma$ is appended by $\{\tilde{h}''_{l,j}(i)\}_{P_l\in\mathcal{P}, 1\le j\le n}$ and $\{\tilde{H}''_j(i) = (\sum_{P_l\in\mathcal{P}} \lambda_l \tilde{h}''_{l,j})(i)\}_{1\le j\le n}$.
    4) In the final SS-Add round, for each $P_i \in \Gamma$, $\mathcal{A}$ adds the input shares to get $\{\tilde{g}'_j(i) = (\tilde{H}''_j + p'_{j,1})(i)\}_{1\le j\le n}$ (i.e., the shares of $\{F(x_j, S'_B) = r_j \tilde{r}'_j f'(x_j) + x_j\}_{1\le j\le n}$), and append to view of $P_i \in \Gamma$.
  - Simulating the *output reconstruction* phase: $\mathcal{A}$ adds the shares $\{\tilde{g}'_j(1), ..., \tilde{g}'_j(N)\}_{1\le j\le n}$ to the view of $A \in \Gamma$. Finally, for party A, $\mathcal{A}$ computes $\mathcal{I}'_{A,B} = S_A \bigcap \{\tilde{g}'_j(0)\}_{1\le j\le n}$. Now $\mathcal{A}$ outputs the views of the corrupted parties and halts.

Next, we show by induction that the partial view (of any length) of parties in Γ in a real execution is identically distributed to the partial view (of the same length) output by the simulator. We start with the base case, after the input stage has concluded.

Claim 1: For every $\vec{x} \in \mathcal{F}_p^{m+n}$ and every $\Gamma \in \mathcal{P}$ with $|\Gamma| \le t$, we have

$$\{\text{VIEW}_\Gamma^{\Pi_1,0}(\vec{x})\} \equiv \{\mathcal{A}^0(\Gamma, \vec{x}_\Gamma, F_\Gamma(\vec{x}))\}. \tag{4}$$

Proof: Note that a general version of this claim where every party has an input has been proven in [26] as Claim 4.3. In our case, only A and B have inputs. Thus, it can be regarded as a special case. For all $P_i \in \Gamma$, they receive shares of values $\{a'_k\}_{1\le k\le m-1}$ and $\{\tilde{r}'_j\}_{1\le j\le n}$ from party $B \notin \Gamma$. Because these shares are calculated from random polynomials $q'_k(x) \in \mathcal{P}^{a'_k,t} (1 \le k \le m-1)$ and $q''_j(x) \in \mathcal{P}^{\tilde{r}'_j,t} (1 \le j \le n)$, respectively, and $P_i \in \Gamma$ only hold less than or equal to t shares of each value, by Claim 3.4 in [26], we have

$$\Big\{\{q'_k(i)\}_{1\le k\le m-1}, \{q''_j(i)\}_{1\le j\le n}\Big\}_{i\in\Gamma} \equiv \Big\{\{Q'_k(i)\}_{1\le k\le m-1}, \{Q''_j(i)\}_{1\le j\le n}\Big\}_{i\in\Gamma}, \tag{5}$$

where $Q'_k(x) \in \mathcal{P}^{a_k,t} (1 \le k \le m-1)$ and $Q''_j(x) \in \mathcal{P}^{r_j,t} (1 \le j \le n)$ stand for the corresponding polynomials chosen by B in the real execution. In addition, the views of $P_i \in \Gamma$ for all simulated shares from their own inputs are indistinguishable from the real execution. Thus, the real and simulated views of the input phase are identical. ■

Claim 2: Let $k \in 1, ..., L$ where L is the number of rounds in Π_1. If

$$\{\text{VIEW}_\Gamma^{\Pi_1,k-1}(\vec{x})\} \equiv \{\mathcal{A}^{k-1}(\Gamma, \vec{x}_\Gamma, F_\Gamma(\vec{x}))\}, \tag{6}$$

then

$$\{\text{VIEW}_\Gamma^{\Pi_1,k}(\vec{x})\} \equiv \{\mathcal{A}^k(\Gamma, \vec{x}_\Gamma, F_\Gamma(\vec{x}))\}. \tag{7}$$

[4]Note that our protocol requires every party to receive all the messages others sent during one round in order to proceed to the next round. Since we consider semi-honest adversaries, this can be easily implemented by a synchronization message.

[5]For clarity, we also add to the view of each party the output shares it holds after computing in each round [26].

Proof: Let $k \in 1, ..., L$. We separately consider the cases that the current round is a SS-Mul-Add or SS-Add or SS-Mul, which is a circuit for product-sum, addition, multiplication, respectively.

If the k-th round is a addition round, then $\{\text{VIEW}_{\Gamma}^{\Pi_1,k}(\vec{x})\}$ is a deterministic function of $\{\text{VIEW}_{\Gamma}^{\Pi_1,k-1}(\vec{x})\}$, which means the views in the k-th round are extended by the addition of shares appearing in the previous view. Thus, if the views up to the $k-1$-th round is identically distributed, the two views in the k-th round are obviously identically distributed. (Note that in our case each round contains multiple same type of gates executed in parallel; however, this is not a problem because the joint distribution of input shares are assumed identically distributed).

If the k-th round is a multiplication round that multiplies shares of a and b, in the real execution the appended view is t shares of $\{h'_{l,j}(0) = [a_j]_l[b_j]_l\}_{P_l\in\mathcal{P}}$ and $H'_j(0) = a_j \cdot b_j$, where $H'_j(x) = \sum_{P_l\in\mathcal{P}} \lambda_l h'_{l,j}(x)$, and $H'_j(x)$ is a random polynomial chosen from $\mathcal{P}^{a\cdot b,t}$ because all the parties' chosen $h'_{l,j}(x)$ are random polynomials. In the simulated execution, the appended view is t shares of $\{\tilde{h}'_{l,j}(0) = [a'_j]_l[b'_j]_l\}_{P_l\in\mathcal{P}}$ and $\tilde{H}'_j(0) = a'_j \cdot b'_j$, where $\tilde{H}'_j(x) = \sum_{P_l\in\mathcal{P}} \lambda_l \tilde{h}'_{l,j}(x)$ which is again a random polynomial. For the part of the $\{h'_{l,j}(0)\}$ and $\{\tilde{h}'_{l,j}(0)\}$ shares coming from $P_l \notin \Gamma$, t random shares of $[a_j]_l[b_j]_l$ are identically distributed to t random shares of $[a'_j]_l[b'_j]_l$, given the partial views up to the $k-1$th round. To see why, denote X_{k-1} and Y_{k-1}, $X_{\Delta k}$ and $Y_{\Delta k}$ as the partial views and added views in k-th round for real and ideal execution, respectively. By Claim 3.4 in [26] we have: given any two sets of secrets $\{a_{l,j}\}_{P_l\in\mathcal{P},1\le j\le n}$ and $\{b_{l,j}\}_{P_l\in\mathcal{P},1\le j\le n}$ (even when the secrets are known):

$$\Big\{\{h'_{l,j}(i)\}_{P_l\in\mathcal{P},1\le j\le n,i\in\Gamma}\Big\} \equiv \Big\{\{h'_{l,j}(i)\}_{P_l\in\mathcal{P},1\le j\le n,i\in\Gamma}\Big\}, \tag{8}$$

where $h_{l,j}(x)$ and $h'_{l,j}(x)$ are random polynomials chosen from $\mathcal{P}^{a_{l,j},t}$ and $\mathcal{P}^{b_{l,j},t}$, respectively. In addition, these random polynomials are chosen independently from the previous partial views X_{k-1} and Y_{k-1}. This means the conditional probability distributions $\Pr[Y_{\Delta k}|Y_{k-1}]$ and $\Pr[X_{\Delta k}|X_{k-1}]$ are indistinguishable. From assumption we have $X_{k-1} \equiv Y_{k-1}$, thus, $X_k \equiv Y_k$.

If the k-th round is a product-sum round that computes the multiplication of $m-1$ pairs of numbers $a_1, b_1, ..., a_{m-1}, b_{m-1}$ and then add all of them together, the similar argument can be made to the above because the only difference with single multiplication gate is, the parties first locally computes the $m-2$ additions and then re-shares. Here, $\{h'_{l,j}(0) = \sum_{k=1}^{m-1}[a_{j,k}]_l[b_{j,k}]_l\}_{P_l\in\mathcal{P}}$ while $\{\tilde{h}'_{l,j}(0) = \sum_{k=1}^{m-1}[a'_{j,k}]_l[b'_{j,k}]_l\}_{P_l\in\mathcal{P}}$. It can be easily seen the rest of the analysis does not change from the above. This completes the proof of the claim. ■

For the output reconstruction stage: the output values of party A in the real and simulated executions are: $\{F(x_j, S_B) = r_j r'_j f(x_j) + x_j\}_{1\le j\le n}$ and $\{F(x_j, S'_B) = r_j \tilde{r}'_j f'(x_j) + x_j\}_{1\le j\le n}$, respectively. In addition, A obtains the random polynomials $\{g'_j(x) \in \mathcal{P}^{F(x_j,S_B),t}\}_{1\le j\le n}$ and $\{\tilde{g}'_j(x) \in \mathcal{P}^{F(x_j,S'_B),t}\}_{1\le j\le n}$ from all parties' shares in the corresponding views. It is easy to see that the output values are identically distributed, because r'_j and $\tilde{r}'_j$ are uniformly chosen by B and the simulator (respectively), while the partial views of Γ after all the previous computations are identically distributed by our induction, which does not give the adversary any advantage in distinguishing r'_j and $\tilde{r}'_j$. Thus, $\{g'_j(x), F(x_j, S_B)\}_{1\le j\le n}$ and $\{\tilde{g}'_j(x), F(x_j, S'_B)\}_{1\le j\le n}$ are also identically distributed given the partial views before the reconstruction stage, which means the complete views are identically distributed.

For the case when $A \notin \Gamma$, $B \in \Gamma$: The behavior of the simulator for Γ is mostly symmetric to the simulator for $\Gamma \ni A$. The only differences in this case are:

(1) The inputs to $\mathcal{A}$ is now $(S_B = \{y_1, ..., y_m\}, 1^\kappa)$ and $\mathcal{I}_{A,B}$.
(2) From the input sharing phase until the end of computation phase, it switches the role from simulating the view of $\Gamma \ni A$ (in case 1)) to $\Gamma \ni B$. The partial views for these phases are again identically distributed, as in the input sharing phase Γ also receives less or equal to t shares, while in the computation phase Γ's views are symmetric to that in case 1).
(3) In the output reconstruction phase, the output values of B in the real and simulated executions are: $\{F(x_j, S_B) = r_j r'_j f(x_j) + x_j\}_{1\le j\le n}$ and $\{F(x'_j, S_B) = \tilde{r}_j r'_j f(x_j) + x'_j\}_{1\le j\le n}$, respectively. For B, the numbers r_j and $\tilde{r}_j$ are identically and uniformly distributed, conditioned on the real/simulated partial views, respectively. Thus, the complete views are identically distributed.

Finally, for the case when $(A, B) \notin \Gamma$, the parties in Γ do not receive any inputs/outputs and only sees less than or equal to t shares throughout the protocol, thus it is easy to show that the complete views are identically distributed in this case using induction. This completes our proof of the theorem. ■

Appendix B
Detailed Proof of Security of our Advanced Protocol Under the HBC Model

Theorem 4: Assuming a secure pseudorandom permutation and semantically secure additive homomorphic encryption scheme, the advanced protocol (in Fig. 2) t-privately computes the cardinality of set intersection between two parties A and B (in the semi-honest model), against computationally bounded adversaries.

Proof: We describe a simulator $\mathcal{A}$ that simulates the adversary Γ's view. We first focus on the case for $A \in \Gamma, B \notin \Gamma$. The simulator proceeds as follows.

- **Input**: The simulator receives inputs and outputs, $(S_A = \{x_1, ..., x_n\}, 1^\kappa)$ and $|\mathcal{I}_{A,B}|$, respectively, from $A \in \Gamma$.
- **Simulation**:
  - Simulating the *input sharing* phase:

  (1) For party A, $\mathcal{A}$ does the same as in the previous proof, except that it first randomly permutes set S_A.
  (2) For party B, it sets the intersection set $\mathcal{I}_{A,B} = (x_1, ..., x_{|\mathcal{I}_{A,B}|})$. The remaining steps in this phase are the same as in the previous proof.

– Simulating the *share-based computation* phase: $\mathcal{A}$ proceeds in the same way as in the previous proof, except without the final SS-Mul round. Recall that in the final SS-Mul round, the view of $P_i \in \Gamma$ is appended by $\{\tilde{h}''_{l,j}(i)\}_{P_l \in \mathcal{P}, 1\leq j\leq n}$ and $\{\tilde{H}''_j(i) = (\sum_{P_l\in\mathcal{P}} \lambda_l \tilde{h}''_{l,j})(i)\}_{1\leq j\leq n}$ which are the shares of $\{F(x_j, S'_B) = r_j \tilde{r}'_j f'(x_j)\}_{1\leq j\leq n}$.

– Simulating the *output reconstruction* phase:

(1) For the *share conversion* round: for each $P_l \in \mathcal{P}$, $\mathcal{A}$ generates random 1-degree polynomials $\{\tilde{u}_{l,j}(x) \in \mathcal{P}^{[F(x_j, S'_B)]_l, 1}\}_{1\leq j\leq n}$, and A's view is appended by $\{\tilde{u}_{l,j}(A)\}_{P_l\in\mathcal{P}, 1\leq j\leq n}$ and $\{\tilde{U}_j(A) = (\sum_{P_l\in\mathcal{P}} \gamma_l \tilde{u}_{l,j})(A)\}_{1\leq j\leq n}$ where γ_l are Lagrangian coefficients.

(2) For the *blind-and-permute* round: $\mathcal{A}$ generates a pseudorandom permutation $\tilde{\pi}$ and uniformly chosen random numbers $\{\tilde{r}''_j\}_{1\leq j\leq n}$. It computes $\{\tilde{U}'_j(A) = \tilde{U}_j(A) + \tilde{r}''_j\}_{1\leq j\leq n}$ and $\{\tilde{U}'_j(B) = \tilde{U}_j(B) - \gamma_A/\gamma_B \tilde{r}''_j\}_{1\leq j\leq n}$, and then permutes both $\{\tilde{U}'_j(A)\}_{1\leq j\leq n}$ and $\{\tilde{U}'_j(B)\}_{1\leq j\leq n}$ using $\tilde{\pi}$. It adds $\{\tilde{U}'_{\tilde{\pi}(j)}(A)\}_{1\leq j\leq n}$ to the view of Γ.

(3) Finally, $\mathcal{A}$ adds the shares $\{\tilde{U}'_{\tilde{\pi}(j)}(B)\}_{1\leq j\leq n}$ to the view of Γ. $\mathcal{A}$ then computes $\{\tilde{U}'_j(0)\}_{1\leq j\leq n}$, and sets $|\mathcal{I}_{A,B}|$ to be the number of zero elements in $\{\tilde{U}'_j(0)\}_{1\leq j\leq n}$. Now $\mathcal{A}$ outputs the views of the corrupted parties and halts.

Next we show that the real/ideal views are computationally indistinguishable. We still use the induction method as in the previous proof. Essentially, the proof for the *input sharing* and *share-based computation* phases remains the same. Thus the partial views of Γ after the computation phase are identically distributed.

For the *share conversion* round, in the real/simulated executions, A's appended view consists of only one share of each $\{u_{l,j}(A)\}_{P_l\in\mathcal{P},1\leq j\leq n}$ and $\{\tilde{u}_{l,j}(A)\}_{P_l\in\mathcal{P},1\leq j\leq n}$, respectively. Since $\{u_{l,j}(x)\}_{P_l\in\mathcal{P},1\leq j\leq n}$ and $\{\tilde{u}_{l,j}(x)\}_{P_l\in\mathcal{P},1\leq j\leq n}$ are all 1-degree random polynomials, these shares are actually identically distributed given their previous respective partial views (using the same line of reasoning as in Eq. (8)).

For the *blind-and-permute* round, in the real/simulated executions, A's appended view consists of $\{U'_{\pi(j)}(A)\}_{1\leq j\leq n}$ and $\{\tilde{U}'_{\tilde{\pi}(j)}(A)\}_{1\leq j\leq n}$, respectively. Since the random numbers r''_j and $\tilde{r}''_j$ are all identically and uniformly distributed, the randomized shares $\{U'_j(A)\}_{1\leq j\leq n}$, and $\{\tilde{U}'_j(A)\}_{1\leq j\leq n}$ are uniformly distributed and independent from the previous real/simulated partial views $X_{k-1} = \{\{U_j(A)\}_{1\leq j\leq n}, ...\}$, and $Y_{k-1} = \{\{\tilde{U}_j(A)\}_{1\leq j\leq n}, ...\}$.

It remains to show that the full views of Γ up to the output reconstruction stage are still computationally indistinguishable. In both views, the output shares received by A define a random polynomial with $\{U'_j(0) = F(x_{\pi(j)}, S_B)\}_{1\leq j\leq n}$ and $\{\tilde{U}'_j(0) = F(x_{\tilde{\pi}(j)}, S'_B)\}_{1\leq j\leq n}$ as the free coefficients, respectively. Due to the security of pseudorandom permutations, the distribution of the positions $(j_1, ..., j_{|\mathcal{I}_{A,B}|})$ and $(j'_1, ..., j'_{|\mathcal{I}_{A,B}|})$ that makes $\{F(x_{j_k}, S_B)\} = 0$ and $\{F(x_{j'_k}, S'_B)\} = 0$ are computationally indistinguishable. For all the other positions containing random values, following the same line of reasoning as in our previous proof, their value distributions are also indistinguishable.

Now we discuss the case for $B \in \Gamma, A \notin \Gamma$. The behavior of the simulator for $\Gamma \ni B$ is mostly symmetric to the simulator for $\Gamma \ni A$. The only differences in this case are:

(1) The inputs to $\mathcal{A}$ is now $(S_B = \{y_1, ..., y_m\}, 1^\kappa)$ and $|\mathcal{I}_{A,B}|$.

(2) The input sharing phase: $\mathcal{A}$ sets S'_A by first choosing a pesudorandom permutation $\pi_1 : \{1, ..., n\} \to \{1, ..., n\}$, and then sets A's input values at $|\mathcal{I}_{A,B}|$ permuted positions to be equal to $|\mathcal{I}_{A,B}|$ of B's inputs. That is, it sets $x_{\pi_1(j)} = y_j, \forall j \in [1, ..., |\mathcal{I}_{A,B}|]$. Other elements in S'_A are set to random numbers (not equal to S_B's elements). The partial views from this phase throughout the computation phase are identically distributed, which have been discussed in proof of Theorem 1.

(3) The *blind-and-permute* round: B receives $\{E_{pk_A}(U_j(A))\}_{1\leq j\leq n}$ and $\{E_{pk_A}(\tilde{U}_j(A))\}_{1\leq j\leq n}$, respectively in the real/simulated views. Due to the semantic security of the additive homomorphic encryption scheme, distributions of $\{E_{pk_A}(U_j(A))\}_{1\leq j\leq n}$ and $\{E_{pk_A}(\tilde{U}_j(A))\}_{1\leq j\leq n}$ are computationally indistinguishable to Γ, while $\{U_j(A)\}_{1\leq j\leq n}$ and $\{\tilde{U}_j(A)\}_{1\leq j\leq n}$ are independent to anything in its corresponding previous partial views. So the partial views up to the final output step are computationally indistinguishable. In the output step, the simulator also computes $\{\tilde{U}'_j(0)\}_{1\leq j\leq n}$, and sets $|\mathcal{I}_{A,B}|$ to be the number of zero elements in $\{\tilde{U}'_j(0)\}_{1\leq j\leq n}$.

(4) Now, we show the full views of Γ are computationally indistinguishable. The distribution of the zero positions $(j_1, ..., j_{|\mathcal{I}_{A,B}|})$ and $(j'_1, ..., j'_{|\mathcal{I}_{A,B}|})$ are indistinguishable, even given all the previous partial views of $\Gamma \ni B$. This is because: (1) the two distributions are independent of B's own input S_B as $f(y)$ does not depend on y_j's positions; (2) the real/simulated positions of $\mathcal{I}_{A,B}$ elements in S_A and S'_A (which B can recover using π and $\tilde{\pi}$) are both generated from pseudorandom permutations, which are computationally indistinguishable by Γ. These are the only information in the view that is related to the distribution of the zero positions for B.

For the case when $(A, B) \notin \Gamma$, the same proof for Theorem 1 applies here because they do not receive any inputs/outputs. This completes our proof. ■