RVClose: Discovering Nearby-Friend Groups in Multi-user Environment (Invited Paper)

Goutam Paul, *Senior Member, IEEE*, Shashwat Raizada, and Vineet Pandey

***Abstract*—In the context of location based services, often a user needs to learn the location of a friend if the friend is in the vicinity. If the friend is outside the specified neighbourhood, then the user is not supposed to know the location of the friend. This is the Nearby-Friend discovery problem. All the existing works either address the problem for two users only, or they do not scale when applied to multi-user setting. In this paper, we for the first time present an effective and yet computationally efficient solution that caters to multiple users. We use techniques from Euclidean plane geometry and graph theory to develop a secure multi-party computation approach whereby each user can determine the respective group of nearby-friends without much computational overhead. The proposed solution ensures users' location privacy by enforcing k-anonymity. Further, we present an application of the underlying logic for two users that act as provers and a semi-trusted third-party that acts as verifier.**

***Index Terms*—k-anonymity, Location privacy, Multi-party Computation, Nearby-friend.**

I. Introduction

Consider the following scenario. In a troubled war zone, Private Ryan's platoon is part of an international peace-keeping operation. His operation involves patrols into zones involving armed conflict against hostile ethnic groups. In one such mission his platoon is trapped, ambushed and under heavy enemy fire. Private Ryan besides facing the enemy fire, must organize casualty evacuation, immediately warn other soldiers *in vicinity*, and also seek their assistance as they form the closest reinforcements. In such a situation if he broadcasts the emergency message, this information will be disseminated across the entire force and will be received by a large number of units that are in no way involved with the skirmish. Such messages are believed to create confusion besides adding to the communication overhead. He thus needs a system that will keep track of the locations of all friendly units and inform him of the nearby ones only. Given the possibility of being captured by the enemy it is unsafe for him to know the location of all units. Such paradoxical requirement of location data sharing yet ensuring location privacy is the motivation for nearby-friend discovery protocols.

Convergence of media has enabled the embedding of location finding technologies like Global Positioning System (GPS) into mobile devices, paving the way for location based services that have a paradoxical requirement of knowing a user's location yet ensuring the user's privacy. The *Nearby-Friend problem* [25] is one such application that informs a user of all its friend in proximity without revealing any information of other friends. Existing works in public domain cater only for two users. We propose a solution for multiple users using *location cloaking* wherein a user's location is obscured by revealing partial location information so that even with the best approximation, an adversary cannot estimate it to less than k many distinct probable points of presence making the user *k-anonymous*. A single trusted location broker is required in the protocol. Partial components of an entity's location are revealed and used by all users to compute the final solution in a secure-multi party computation fashion.

If we consider the set $\mathcal{S}$ of users $\{U_1, U_2, \ldots, U_n\}$, then for a user U_i, the *multi-party nearby-friend problem* requires forming the set of users g_i such that
$g_i = \{f_{i1}, f_{i2}, \ldots, f_{in}\}, 1 \leq i \leq n$, where $f_{ij} = 1$ *if* U_i and U_j are nearby, *else* $f_{ij} = 0$.

A. Related Works

While much of the existing works cater to proximity testing of two users at a time, our work is first of its own kind that extends the principle of discovering multiple nearby friends at a given time by a single user, without much computational overhead.

Zhong et al. [25] proposed the Louis, Lester and Pierre protocol for finding nearby-friends in a two party scenario. Gupta et al. [6] have claimed to have found flaws in the Louis protocol and proposed certain modifications. Liu and Gedik [5] have introduced *k-anonymity* using spatial and temporal cloaking and provided customizable anonymity levels, a feature of our work as well. Chatterjee et al. [1] have proposed another protocol resembling a Diffie-Hellman type of key exchange [3].

A number of smart-phone applications have sprung towards providing location based services, but their privacy considerations are unavailable in open literature. Google and CMU had

This is a revised and extended version of the paper "Nearby-Friend Discovery Protocol for Multiple Users" that was presented in PASSAT 2009. We have revised and rewritten Section I-A (Related Works) with new state-of-the-art research results that have appeared after the publication of the above conference version up to the very recent time. Further, Lemma 3.1 and Theorem 3.2 are new additions in the current version.

G. Paul is with the Cryptology and Security Research Unit (CSRU), R. C. Bose Centre for Cryptology and Security, Indian Statistical Institute, Kolkata 700 108, India. Email: goutam.paul@isical.ac.in

S. Raizada is with the Indian Statistical Institute, Kolkata 700 108, India. Email: swiftsr@gmail.com

V. Pandey is with Department of Computer Science and Engineering, University of California, San Diego, USA. Email: vineetp13@gmail.com

joined the fray, with their Latitude [11] and Locaccino [16] services which were direct implementations of nearby-friend, but both were shut down in 2013. On the other hand, "Find My Friends" is an application and service for iOS developed by Apple Inc. in 2011 and so far it has survived several version updates.

A recent work [8] addresses proximity-based privacy challenge for multiple users. It is based on geo-indistinguishability, differential privacy principles and encryption techniques. While, it is referred to as "honest-but-curious" service provider (SP), yet the user shares its generated symmetric key with the SP which can be of potential risk. Our work on the contrary, does not share critical information with the server that could be vulnerable in the long run.

Currently, much work revolve around the use of Homomorphic encryption techniques and Paillier cryptosystem to address nearby friend discovery problems (citeMuyuan,[19],[7],[21],[4]). However, the techniques are computationally expensive and are not suitable for resource-constrained devices. Ciphertexts in homomorphic encryptions are much larger than the plaintexts, thereby increasing communication overhead. Given our scenario of multiple parties, such mechanism shall require high computational power. Our technique uses a very simple yet effective technique to address the challenge.

In [2], the technique utilises WiFi Direct Technology and thus is not be feasible for non-WiFi users as well as in domains where there exists no WiFi. Further, it requires intensive key management schemes to address a multiple parties scenario. Our mechanism on the other hand, is feasible in varying network architecture and caters to multiple parties without hassles of a key management scheme. In [14], the users must reveal the slope that are relevant to zones they belong to. Such disclosure has potential risk. Our technique reveals only the Hash value of proximity distance along with a user ID (which is globally known) for necessary computation. Such information sharing is not vulnerable. [15], provides a protocol for device discovery and recognition via short range radio. Again, such mechanism is unlikely to cater to a large span of area and key management will need to be extensive to cater to multiple parties. [24], use location tagging for proximity test, however extraction of location tags from packets have risks of information leakage. [20] uses key management mechanisms for proximity testing. Such mechanisms again, need elaborate and scalable architecture to sustain multiple parties interaction.

II. Solution Implementation Scenario

Like other protocols in this area, services like authentication and secure communication channel are presumed to be existing a priori.

The solution considers the implementation topology to be a two-dimensional grid, where zones are represented by squares. The geometric center coordinates (a pair of the form of the square are used for computation by users located anywhere in the square.

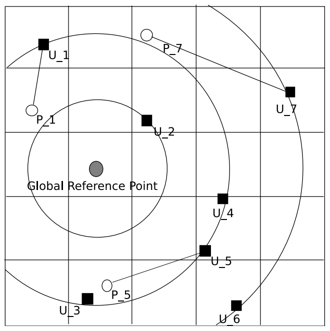

Fig. 1. Grid overview

A. Definitions and Notations

1) Grid overview: The following terms are used in the paper:

- *Grid*: Refers to the deployment area modeled as a two-dimensional plane that is divided into squares of equal area. All users are located in this grid.
- *Center-point*: The geometric center of a square in the grid is its center-point which represents any point inside that square. Hence all points in a particular square are considered to be equivalent.
- *Nearby region for a user*: The square occupied by the user.
- *Nearby-friend*: U_j is said to be a nearby-friend of U_i if they both lie in the same square.
- *Global Fixed Reference point*: A point P_g which is the same for all the users.
- *User Chosen Reference point*: A point P_j which is uniquely determined for each U_j.

2) Anonymity Factors:

- *k-anonymity index*: A value chosen by each user U_i which provides for privacy protection by guaranteeing that with the amount of location information revealed, the adversary will only be able to approximate its position to at least k points.
- *Privacy Factor*: Denoted by m, this value is used for forming coarse grain nearby-user groups.

Fig. 1 gives an overview of the grid structure.

3) Notation:

- S_i^1 : Coarse Grain Nearby Users (Potential nearby-friends of user U_i after *preprocess* step).
- S_i^2 : Fine Grain Nearby Users (Potential nearby-friends of user U_i after *refine* step).
- S_i^3 : Final Nearby Users (Nearby-friends of user U_i after *determine* step).
- s_i : Random salt as selected by user U_i.
- d_i^g : Distance of User U_i from *global reference point* P_g.
- d_i^j : Distance of User U_i from Point P_j.
- q_i : The order of user U_i which is equal to $\lceil d_i^g/m \rceil$.
- r_i^g : Modular distance of User U_i from Point P_g calculated as $d_i^g \bmod m$.
- T_S : Semi Trusted Third-party.
- $\mathcal{H} : \{0,1\}^* \longrightarrow \{0,1\}^l$: Hash function.
- $\mathcal{I}_i$: Identity of user U_i.

B. Secure Distance Comparison Function

Secure distance comparison solves the following problem: two users U_i and U_j are at distances val_i and val_j respectively from a point P_i selected by U_i. Given the coordinates of P_i, the user U_j must ascertain whether U_i and itself are equidistant from P_i, i.e., whether val_i is equal to val_j without learning the location of U_i.

The following procedure is adapted. User U_i generates a random salt s_i and computes $\mathcal{H}(s_i, \mathcal{I}_i, val_i)$. U_i next transmits P_i, s_i and $\mathcal{H}(s_i, \mathcal{I}_i, val_i)$ to U_j who in turn computes its distance val_j from P_i and computes $\mathcal{H}(s_i, \mathcal{I}_i, val_j)$. The two users are equidistant from P_i if $\mathcal{H}(s_i, \mathcal{I}_i, val_i) = \mathcal{H}(s_i, \mathcal{I}_i, val_j)$

```
$U_i \longrightarrow U_j$: $s_i, P_i, \mathcal{H}(s_i, \mathcal{I}_i, val_i)$
$U_j$: computes $\mathcal{H}(s_i, \mathcal{I}_i, val_j)$
if($\mathcal{H}(s_i, \mathcal{I}_i, val_i) = \mathcal{H}(s_i, \mathcal{I}_i, val_j)$)
    say $equidistant$ else $notclose$
```

User U_j only learns whether U_i and itself are equidistant from P_i in the event of the two $\mathcal{H}$ values being the same. However, if the two $\mathcal{H}$ values are not same, it becomes explicitly clear that U_j and U_i cannot be nearby-friends. In such a case, U_j is not privy to val_i. The only way it can compute val_i is by finding the pre-image of the hash value. The random salt s_i has been introduced to thwart a table look-up type of attack. Despite all these measures, even if the adversary is able to find val_i the protocol is built to account for k-*anonymity*.

III. RVClose Protocol

The protocol derives its name from 'RV', which is the military-acronym of 'rendezvous'; it may also be read as a question "Are We Close?". Thus the name signifies identification of proximity for a potential meeting.

A. Main Idea

A point on a plane can be precisely determined by its distance from three non-collinear known points using triangulation technique. Thus, co-located users are equidistant from three non-collinear reference points. In this protocol, each party indicates its distance from a distinct point and the other users check whether their distances from that point are the same and accordingly inform the semi-trusted third-party. Two users informing the third-party of a match in their distances, with reference to the points indicated by each other, is indicative of the fact that they might be nearby. Geometrically the users will be at the points of intersection of two circles drawn using the fixed points as the center and the radial distance from the user as the radius. Such circles intersect in at-most two points and so the users may be either close-by or at alternate points of intersection (not close-by) and a dual point resolution mechanism is used to solve the ambiguity. Thus of the three radial distances used to determine a location, the first is used to obtain a course-grain identification of nearby-friends, while the second and third are used intuitively to establish proximity. An algorithm based on graph theory for finding maximal cliques is presented to ensure polynomial time computation at the trusted third party.

B. Protocol Description

Protocol has two distinct implementation phases, first being *protocol initialization*, in which the users and the semi-trusted third-party form groups of nearby users followed by *protocol operations* phase, which caters to changes in existing user's motion and entry of new users.

C. Protocol Initialization

The Protocol Initialization consists of the following three steps:
Preprocess step: all the users form a coarse grain nearby-friend set S_i^1, after matching modular distances from a fixed point P_g.
Refine step: each user U_i processes S_i^1 to form a fine grain nearby-friend set S_i^2 consisting of users U_j which have the same distance from P_j as U_i.
Determine step: T_S receives S_i^2 from each user U_i and computes the final nearby-friend group S_i^3.

1) Preprocess: There is a *global reference point* P_g and a *privacy factor* m for a given region in which all the users are present. The user U_i calculates its distance from P_g as d_i^g and then calculates its r_i^g and q_i values using the formula:

$$\boxed{r_i^g = d_i^g \bmod m \text{ and } q_i = \lceil d_i^g/m \rceil}$$

He broadcasts the value r_i^g to all users.

$$\boxed{U_i \longrightarrow \text{All users } U_j\text{: } r_i^g}$$

Similarly, the user U_i receives such values as broadcasted by other users, and compares the received r_j^g values with its r_i^g value for a match. The matched values form a coarse grain user-set S_i^1 of the possible nearby-friends of user U_i. The advantage of preprocessing is that it reduces the number of potential nearby-friends. User U_i carries out the next stages of the protocol only for this set of users of size η, which is less than n.

2) Refine: After the preprocessing step, each user forms its coarse grain nearby user set S_i^1 consisting of users with the same r_i^g. Every user U_i selects a User Chosen Reference point P_i at a distance $\geq (k_i m)/4$ (Theorem 3.1 establishes how k-*anonymity* is ensured) and sends this to all users U_j in S_i^1, and receives their P_j values. U_i then checks if any P_j, P_i and *global reference point* P_g are lying on the same line (three points p_1, p_2, p_3 are collinear if and only if the $slope(p_1, p_2)$ = $slope(p_1, p_3)$). If this is the case, then U_i changes its point P_i and sends this to all users in S_i^1. User U_i computes its distance d_i^i from P_i, chooses a random salt s_i and initiates the secure distance comparison function with other members of S_i^1 with $val_i = d_i^i$. It then receives the respective data from all other users U_j in S_i^1.

$$\boxed{U_i \longrightarrow S_i^1\text{: } (P_i, s_i, \mathcal{H}(s_i, \mathcal{I}_i, val_i))}$$

Every user U_j in S_i^1 who gets a match in the secure distance comparison function with the values of U_i, adds U_i to its S_j^2. Similarly, U_i creates its set of fine grain nearby users S_i^2. This forms the basis of secure multi-party computation where all the players hold private inputs (whether other users are potential

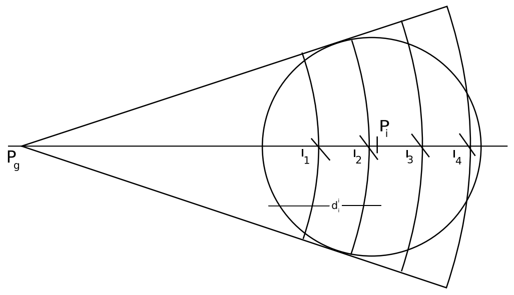

Fig. 2. Illustrating Theorem 3.1

nearby-friends) and want to further ascertain whether they are actually nearby.

Introducing anonymity: k-*anonymity* refers to an item being indistinguishable from (k-1) other items.

Theorem 3.1: With the knowledge of the points P_g, P_i along with the values r_i^g, d_i^i pertaining user U_i, one can estimate its position to not less than $k = (4d_i^i)/m$ locations.

Proof: In Fig. 2, d_i^i is the radius of the circle centered at P_i. There is a set of concentric circles $\mathcal{C}$ with center at P_g each with radius r that satisfies equation $r_i^g = r \bmod m$. Thus the distance between successive circles concentric circle is equal to $privacy factor\ m$. I_1 to I_4 are the intersections of the diameter (on the line joining the centers) with the circles in $\mathcal{C}$. Distance between two consecutive intersections on the diameter is m and so the number of such intersections is $(2d_i^i)/m$. Hence, the number of intersections on the circle perimeter would be twice this number, which should be equal to the anonymity factor k. ∎

Thus, a user U_i, seeking to be k anonymous, will have to select its point P_i at a distance $\geq (k_i m)/4$.

Every user U_i knows that itself and U_j ($\in S_i^2$) are equidistant from P_j, which means that they lie on a circle with P_j as the center and radius = d_i^j. Since r_i^g = r_j^g from pre processing step, the two users can be located anywhere on the circles with radius = αm +r_i^g, where $\alpha \in \{0, 1, 2, \dots\}$.

The intersection of this set of concentric circles with the circle centered at P_j gives at least k points, where the value of k is decided by the user (from Theorem 3.1). Hence the position of U_i or U_j on the circle is obscured in k-*anonymity* for the other. For example, in Fig. 2, there are 8 such points. At the end of *refine* step, U_i sends the set S_i^2 to T_S.

$$\boxed{U_i \longrightarrow T_S\text{: } S_i^2}$$

3) Determine: The semi trusted third-party T_S has now received fine grain nearby-friend sets S_i^2 of all users. Thus for each user, T_S knows the list of users which might be in close proximity to them. We find a set $\mathcal{U}$ of all users U_k who are present in both S_i^2 and S_j^2 and hence potential nearby-friends of U_i and U_j. We then proceed to check if all users U_k in $\mathcal{U}$ are nearby-friends using their S_k^2. To extend this to multiple users, we apply a graph-theoretic approach described below.

Finding groups of nearby-friends: T_S constructs an undirected graph G, where nodes represent users and an edge connects U_i and U_j if and only if $U_i \in S_j^2$ and $U_j \in S_i^2$. Each group of nearby-friends forms a maximal clique. In general, finding all cliques of a graph is an NP-complete problem [10]. Fortunately, the graph G turns out to be a special graph for which we can devise an efficient algorithm, which we call *FindFriendCliques*. As per our grid structure, a user cannot belong to two different nearby-friend groups simultaneously. So in our graphs, no two maximal cliques share a common edge. Each non-clique edge (we call it a *false edge*) either connects two maximal cliques or a maximal clique with a single vertex which is not a member of any nearby-friend group. The main idea is to color all edges *black* initially; and at each iteration, randomly pick a *black* edge and find the maximal clique in which this edge belongs. If we are able to determine such a clique, then we colour the clique edges *white*. If not, we let the edge be coloured *grey*. Also, all edges incident on a clique from vertices lying outside the clique are labelled *red* signifying *false* edges. At the end of the algorithm, *grey* edges denote clique edges for isolated cliques of size 2 and *white* edges denote clique edges for cliques of size more than 2.

For any vertex v, let $N(v)$ denote the neighbouring vertices of v.

```
Input: Graph $G$ constructed by $T_S$.
Output: All maximal cliques in the graph.
(Initialization) Mark all edges as $black$;
while count of black edges $> 0$ do
    Arbitrarily select a black edge $e = \{v_a, v_b\}$;
    Set $I_{ab}$ = $N(v_a) \cap N(v_b)$;
    if $I_{ab}$ is empty then
        Mark the edge $e$ as $grey$;
    else
        Form Maximal clique $C$ =
        $I_{ab} \cup \{v_a\} \cup \{v_b\}$;
        Mark $e$ as $white$, $\forall$ $e \in C$;
        Mark $e'$ as $red$, $\forall$ $e'$ incident on $C$;
    end
end
```

Algorithm 1: FindFriendCliques

Lemma 3.1: During the execution of *FindFrndCliques*, the set I_{ab} always forms a clique.

Proof: If I_{ab} is empty or has only one node, then I_{ab} is trivially a clique. So, we can assume that $|I_{ab}| >= 2$. Suppose I_{ab} contains two nodes x and y such that there is no edge between x and y. Thus, nodes v_a, v_b and x are part of some maximal clique C_1, while nodes v_a, v_b and y are part of a different maximal clique C_2. (Since there is no edge between x and y, they can't be part of the same clique.) This is a contradiction since cliques C_1 and C_2 share the edge $\{v_a, v_b\}$. (Recall that the graph G has the property that no two maximal cliques can share an edge.) Therefore, the nodes in I_{ab} must form a clique. ∎

Theorem 3.2: The worst case time complexity of *FindFriendCliques* algorithm is $\mathcal{O}(n^2)$, where n is the number of users.

Proof: We assume that the graph is stored as an $n \times n$ adjacency matrix. Then finding the intersection takes $\mathcal{O}(n)$ time. Moreover, from Lemma 3.1, I_{ab} always forms a clique. Thus, the number of iterations of the while loop is $\mathcal{O}(n)$ and each iteration takes only $\mathcal{O}(n)$ time, giving the algorithm a worst case running time of $\mathcal{O}(n^2)$. ∎

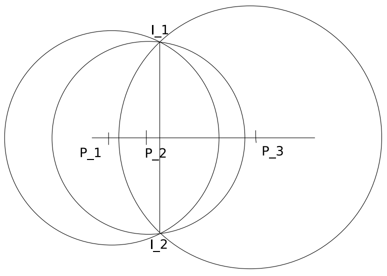

Fig. 3. Illustrating Lemma 3.2

Dual point confusion resolution: Circles centered at points P_i and P_j intersect on at most two points with the possibility of U_i and U_j lying on alternate points of intersection. Due to this *dual-point confusion* for all cliques of size 2 there is a chance of false alarms being raised. This does not happen for a clique of size > 2 as three or more user defined points uniquely identify a location. For two users U_i and U_j in a clique of size 2, we propose the following correction using their distances d_i^g and d_j^g from global reference point P_g.
Different d_i^g: If $d_i^g \neq d_j^g$, then U_i and U_j lie at different distance from P_g. Since $r_i^g = r_j^g$, this can be checked using the order value q_i. If $q_i = q_j$, then U_i and U_j are nearby-friends, else not.
Same d_i^g: For $d_i^g = d_j^g$, users may be nearby. We preempt this problem by using the collinearity condition while choosing the user chosen reference points. If users located at distinct locations are equidistant from three defined points, then these points would be collinear following Corollary 3.1.

Lemma 3.2: If a circle passes through the intersections of two given circles, then the centers of the three circles have to lie on the same line.

Proof: Refer to Fig. 3. Circles C_1 and C_2 intersect at points I_1 and I_2. Hence the line joining P_1 and P_2 is the perpendicular bisector of the common chord I_1I_2. If a third circle C_3 passes through I_1 and I_2, then the line joining P_3 and P_1 would also be the perpendicular bisector chord I_1I_2. Since the perpendicular bisector of a line segment is unique, the result follows. ■

Corollary 3.1: The centers of three circles are collinear if and only if the circles have a total of exactly two intersection points.

T_S generates a random salt s_T and sends it to users U_i and U_j who in turn return $\mathcal{H}(s_T, q_i)$ and $\mathcal{H}(s_T, q_j)$ respectively. If the two values match, then the U_i and U_j are nearby else not. This resolves dual point confusion and S_i^3 is generated as the final solution.

$$\boxed{T_S \longrightarrow U_i\text{: } (S_i^3)}$$

D. Protocol Operations

Post initialization, the existing users are aware of their nearby-friends and the protocol operations commence which comprise of the following tasks.

1) Node Movement: When a node leaves its grid position, it must immediately inform T_S failing which it would imply that the system considers the user to exist at the last known position. T_S broadcasts the ID $\mathcal{I}_j$ of the user U_j that leaves the grid. Users having the ID in their S_i^3 list would remove the user from their list. Such a user must reinitialize itself upon reaching its new grid position.

2) Re-initialization: This is performed when a new node joins the grid or an existing node has moved to a new position. The incoming node U_k computes its r_k^g and sends it to T_S. Then T_S forms a set of potential nearby-friends, by taking one member from each of the established nearby-sets S_i^3 that have the same radius r_k^g. T_S directs the selected members to commence the secure distance comparison by transmitting their respective s_i, P_i and $\mathcal{H}(s_i, \mathcal{I}_i, val_i)$ to U_k. The incoming user U_k checks if a user U_i and itself are equidistant to U_i's user-defined point P_i. If there isn't a match then user U_k does not have any nearby-friend. Upon finding such a match, U_k initiates a secure distance comparison by transmitting s_k, P_k and $\mathcal{H}(s_k, \mathcal{I}_k, val_k)$ to such users. The existing users establish whether point P_k is equidistant to U_k and itself. They report a match to T_S who in turn performs a dual-point confusion check and thereafter, places U_k in the pre-existing set of the matched user. It next informs all members of the group regarding the presence of a new user U_k and also informs U_k about other group members.

In practice if an incoming user U_k spots a friend U_i nearby, it can bypass above steps by informing T_S about it. T_S would ascertain the set S_i^3 of U_i and thereafter deputes one member U_j to commence secure distance comparison with U_k. If U_k and U_j confirm that they are equidistant from P_j and P_k then T_S does a dual point confusion check and includes U_k into the set S_i^3. This communication requires a constant steps and thus the cost of addition of new nodes in the grid is $\mathcal{O}(1)$.

3) Truant Node Detection: If a user U_i spots another node U_t in its vicinity but not in its set S_i^3, it alerts T_S who ascertains whether U_t exists in any other list. If it does, it is classified as a *truant node*, i.e., a node absent from its original position.

IV. Efficiency and Security Analysis

A. Efficiency Analysis

1) Communication: The number of messages broadcasted in preprocessing is n, which results in formation of smaller user-sets of maximum size η (say), where $\eta \ll n$. In *refine* step, each user U_i send its fixed point P_i to S_i^1 in total $n\eta$ messages. After this, the secure distance comparison is performed and it takes n steps. At the end of *refine* step, total n steps are needed to send S_i^2 to T_S. At the end of the next step, T_S informs all users regarding their friend sets in n steps. Hence the total number of steps in our protocol is $3n + n\eta$ which is $\mathcal{O}(n\eta)$. Both the actual number of steps as well as the order is a significant improvement over existing protocols when scaled for multi-user scenario.

$Protocol$	$RVClose$	$Louis$	$Lester$
Third-party	Yes	Yes	-
Communication steps	$3n + n\eta$	$6n^2$	$2n^2$

Communication overhead of different protocols

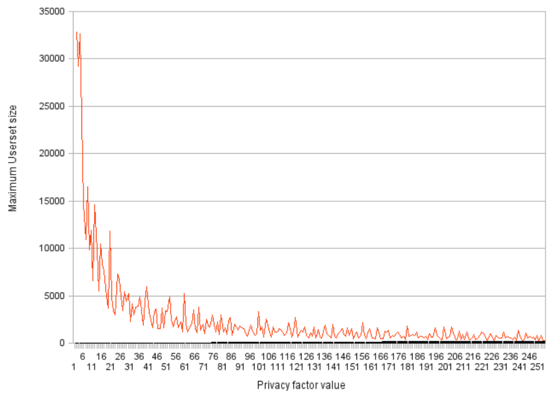

Fig. 4. User-set maximum size (η) vs privacy factor (m)

2) Computation: All nodes perform secure distance computation $n\eta$ times, and check points for collinearity in $\mathcal{O}(\eta)$ time hence the complexity is $\mathcal{O}(n\eta)$. T_S finds maximal cliques in graph G in $\mathcal{O}(n^2)$ time.

B. Privacy factor m and Performance

The privacy factor m must be carefully selected as it largely affects the overall efficiency. A large m yields smaller course grain refinement thereby resulting in lesser computation. However m is constrained by the grid block size. The selection of m is dependent on the number of users and network topology. In our work, we simulated diverse deployment scenarios for n users and were able to build course grain user-sets S_i^1 of maximum size $\eta = n^{\frac{11}{16}}$, which significantly reduces the number of computations for large n. Fig. 4 shows simulation results using 2^{16} users randomly spread over a 256×256 square unit grid. The impact of a change in m, which varied from 1 to 255, on the maximum users-et size η was observed.

When the value of m is very small, user-set size (η) is very high. As the value of m increases, η falls sharply and for $m = 2^7$, $\eta \approx 2500$ as a lone worst case. Hence $\frac{\eta}{n} \approx 0.04$. The sharp fall of user-set size with increase in privacy factor value ensure performance enhancement as the successive calculations by each user would be performed for η users and not n users.

C. Security Analysis

1) Honest Users: Like other privacy protocols, users are considered *honest yet curious* and they only try to learn others' positions using the data legitimately exchanged during the normal protocol working. However, if a user pretends to be at a different grid point, it will form an entirely wrong set of nearby-friends and would risk being classified as a *truant user*. If a truant user attempts to do this at small intervals for different grid, the semi-trusted third-party can observe the pattern and infer that the user is not acting genuinely. In the secure distance computation, user U_i's distance d_i^i from a fixed point P_i in hashed format $\mathcal{H}(s_i, \mathcal{I}_i, d_i^i)$ is known to others. The pre-image of this hash value is essential to obtain d_i^i. Even if d_i^i is known, user U_i continues to be obscured across k anonymous positions.

2) Semi Trusted Third Party: T_S possesses the following information regarding all users: P_g, m, S_i^2 and S_i^3. T_S never receives the actual positions, so it can only form groups of nearby-friends.

3) Adversary: All communication happens between users and T_S over an authenticated and secure channel. An adversary must circumvent the existing authentication and confidentiality schemes by masquerading as a legitimate node, say U_a. Thereafter following the protocol operations U_a can only ascertain users who are its nearby-friend (user-set S_a^1). It however will be caught the moment original U_a enters the protocol.

V. An Application for Two-party Scenario

We elucidate the solution by showing an application for the two-party nearby-friend problem. Here two friends Alice and Bob as *provers* want to establish whether they are nearby with the help of a trusted third-party Trent as *verifier* who must not learn any details. We do not assume the existence of an authenticated and secure environment, hence a public key cryptosystem is used.

A. Notation

$\mathcal{D}_T$	: Decryption by Trent using his private key
s_U	: Random salt generated by user U
P_i	: Random points chosen by Trent
h	: hash function $\{0,1\}^* \longrightarrow \{0,1\}^l$
H_U	: Hashed value for user U
$\mathcal{E}_T$	: Encryption using Trent's public key
$\mathcal{V}_U$	: Value for user U sent over the network
d_i^U	: Distance of user U from point P_i
$\|\|$	: Concatenation function

Alice $\longrightarrow$ Bob: s_A
Bob $\longrightarrow$ Alice: s_B
Trent $\longrightarrow$ Alice, Bob: P_1 P_2
Alice: $H_A = (h(d_1^A + s_A) \,\|\, h(d_2^A + s_B))$
Bob : $H_B = (h(d_2^B + s_B) \,\|\, h(d_1^B + s_A))$
Alice $\longrightarrow$ Trent: $\mathcal{V}_{Alice} = \mathcal{E}_T(H_A)$
Bob $\longrightarrow$ Trent: $\mathcal{V}_{Bob} = \mathcal{E}_T(H_B)$
Trent: $\mathcal{D}_T$ ($\mathcal{V}_{Alice}$)
Trent: $\mathcal{D}_T$ ($\mathcal{V}_{Bob}$)
Trent: Compare (H_A, H_B)
Trent $\longrightarrow$ Alice, Bob: Yes/No

B. Description

In our solution, we have two runs of the protocol. The second run is called for only in case of a match in the first run.

Alice and Bob generate random salts (s_A, s_B) and exchange it with each other. They are next sent the location of two distinct points P_1 and P_2 by Trent. Alice (resp. Bob) calculates her (resp. his) distance from the given points: d_1^A and d_2^A

(resp. d_1^B and d_2^B). Alice then computes $H_A = (h(d_1^A + s_A) \parallel h(d_2^A + s_B))$. Bob does a similar computation with a minor difference. Bob does the concatenation operation in reverse order to generate $H_B = (h(d_2^B + s_B) \parallel h(d_1^B + s_A))$. Alice (resp. Bob) then encrypts this value using Trent's public key and transmits it to Trent who decrypts the received messages and finds H_A and H_B.

Since both Alice and Bob are using the same encryption operation, it is essential to change the plaintext because if Alice is able to read the value being sent by Bob, then she can alter her H_A value to pass the same ciphertext which will be decrypted by Trent to find a match thereby resulting in a false conclusion. Hence, the change in order is necessary to prevent *man in the middle attack* by one of the provers.

Trent proceeds to separate the two hashed values contained in H_A, since it knows the hash family and hence the length of the digest being used. Trent then compares the first hash value contained in H_A with second hash value of H_B and vice versa. If the two values do not match then the users are not collocated. However in case of a match, the two users may not exist at the same point *(as explained further)* and the process would be executed once more to ensure.

In the next run, Trent sends two points P_3 and P_4 which are not collinear with P_1 and P_2. If there is a match at the end of Run II as well, then Alice and Bob are nearby, else not. Though Run II happens whenever Run I generates a match, it is required essentially only when both P_1 and P_2 lie on the perpendicular bisector of distinct locations of Alice and Bob. Since the points P_i are randomly chosen, the probability of this event is very small. In all other cases, Run II becomes a check.

The solution was checked using SHA-256 as hash and 1024 bit RSA as public key cryptosystem. Two 256 bit hash digest were concatenated and encrypted using 1024 bit RSA encryption. This resulted in a single block encryption.

VI. Conclusions

The solution incorporates inputs and calculations from all users and is a secure multi-party computation. Geometry has been intuitively used to establish both co-location and uncertainty. Simple functions like secure distance comparison etc. makes it easy to implement on low end devices. The polynomial time algorithm to locate maximal cliques reduces computation load on T_s. Simulations have demonstrated that the protocol is sufficiently fast to be practical. The operations used are secure and computationally lighter than those of the existing protocols. This work is the first to tackle nearby-friend problem in the multi-user scenario.

References

[1] Sanjit Chatterjee, Koray Karabina and Alfred Menezes. A New Protocol for the Nearby Friend Problem. In *Cryptography and Coding* 2009, LNCS 5921, pages 236–251, Springer.

[2] Eric Chung, Joshua Joy, Mario Gerla. DiscoverFriends: Secure Social Network Communication in Mobile Ad Hoc Networks In *International Wireless Communications and Mobile Computing Conference, IWCMC*, 2015.

[3] Whitfield Diffie and Martin Hellman. New directions in cryptography. In *IEEE Transaction on Information Theory* 22, pages 644–654, 1976.

[4] Marco Furini,Valentina Tamanini. Location privacy and public metadata in social media platforms: attitudes, behaviors and opinions *Multimedia Tools and Applications*,Vol.74, Issue:21, pages 9795–9825, 2015.

[5] Bugra Gedik and Ling Liu. Protecting location privacy with personalized k-anonymity: Architecture and algorithms. In *IEEE Trans. Mob. Comput.*, 7(1):1–18, 2008.

[6] Aakar Gupta, Milan Saini and Anish Mathuria. Security analysis of the louis protocol for location privacy. In *International Conference on COMmunication Systems and NETworkS*, 2009.

[7] Per A. Hallgren, Martn Ochoa, Andrei Sabelfeld. InnerCircle: A parallelizable decentralized privacy-preserving location proximity protocol In *Conference on Privacy, Security and Trust, PST 2015*,pages 1–6, 2015.

[8] Cheng Huang , Rongxing Lu, Hui Zhu, Jun Shao, Abdulrahman Alamer, Xiaodong Lin. EPPD: Efficient and Privacy-Preserving Proximity Testing with Differential Privacy Techniques In *IEEE International Conference on Communications*, 2016.

[9] Ting-Kai Hwang, Yung-Ming Li, Bih-Huang Jin. A Nearby Expert Discovering Mechanism: For Social Support In *Advances in Intelligent Systems and Computing*, Vol.444, pages 943–949, 2016.

[10] Richard M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations, New York: Plenum*, pages 85–103, 1972.

[11] Google Latitude. Location-aware mobile app. In *http://www.google.com/latitude*, Last accessed December 2010.

[12] Muyuan Li, Zhaoyu Gao, Suguo Du, Haojin Zhu, Mianxiong Dong, K. Ota PriMatch: Fairness-aware secure friend discovery protocol in mobile social network In *Global Communications Conference (GLOBECOM), 2012 IEEE*,pages 738–743, 2012.

[13] H. P. Li, H. Hu and J.Xu. Nearby Friend Alert: Location Anonymity in Mobile Geosocial Networks In *IEEE Pervasive Computing*, Vol.12,Issue:4, pages 62–70, 2013.

[14] Xin Lin, Haibo Hu, Hong Ping Lim Jianliang Xu, Byron Choi. Private Proximity Detection and Monitoring with Vicinity Regions In *MobiDE* , pages 5–12, ACM, 2013.

[15] Matthew Lentz, Viktor Erdélyi, Paarijaat Aditya, Elaine Shi, Peter Druschel, Bobby Bhattacharjee. SDDR: Light-Weight, Secure Mobile Encounters In *USENIX Security Symposium (USENIX Security 14)*, pages 925–940, 2014.

[16] CMU Locaccino. A user-controllable location-sharing tool. In *http://www.locaccino.org/*, Last accessed December 2010.

[17] Sehrish Malik, Jong-Hyouk Lee. Privacy Enhancing Factors in People-Nearby Applications In *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA*,Vol.6, pages 113121, 2015.

[18] Marcin Nagy, Thanh Bui, Swapnil Udar, N. Asokan, Jörg Ott. SpotShare and nearbyPeople: applications of the Social PaL framework In *Conference on Security and Privacy in Wireless and Mobile Networks*,pages 31:1–31:2, 2015.

[19] E. Novak, Q. Li. Near-pri: Private, proximity based location sharing In *INFOCOM, 2014 Proceedings IEEE*,pages 37–45, 2014.

[20] E. Oriero, K. Rabieh, M. Mahmoud, M. Ismail, K. Akkaya, E. Serpedin, K. Qaraqe. Trust-Based and Privacy-Preserving Fine-Grained Data Retrieval Scheme for MSNs In *IEEE Wireless Communications and Networking Conference (IEEE WCNC)*,2016.

[21] Constantinos Patsakis, Panayiotis Kotzanikolaou, Mélanie Bouroche. Private Proximity Testing on Steroids: An NTRU-based Protocol In *11th International Workshop,Security and Trust Management (STM) 2015*,pages 172–184, 2015.

[22] Q. Xie, U. Hengartner. Privacy-preserving matchmaking For mobile social networking secure against malicious users In *Privacy, Security and Trust (PST), 2011*, pages 252 – 259, 2011.

[23] M. Xue, Y. Liu, K. W. Ross, H. Qian. I know where you are: Thwarting privacy protection in location-based social discovery services In *IEEE Conference on Computer Communications Workshops(INFOCOM WKSHPS), 2015*, pages 179–184, 2015.

[24] Yao Zheng, Ming Li, Wenjing Lou, Y. Thomas Hou. SHARP: Private Proximity Test and Secure Handshake with Cheat-Proof Location Tags In *European Symposium on Research in Computer Security*, pages 361–378, 2012.

[25] Ge Zhong, Ian Goldberg and Urs Hengartner. Louis, lester and pierre: Three protocols for location privacy. In *Privacy Enhancing Technologies*, pages 62–76, 2007.