# EPPD: Efficient and Privacy-Preserving Proximity Testing with Differential Privacy Techniques

Cheng Huang*†, Rongxing Lu*, Hui Zhu†, Jun Shao‡, Abdulrahman Alamer§, and Xiaodong Lin§
*School of Electrical and Electronic Engineering, Nanyang Technological University, 50 Nanyang Avenue, Singapore
†School of Telecommunications Engineering, Xidian University, Xi'an, China 710126
‡School of Computer and Information Engineering, Zhejiang Gongshang University, P.R.China, 310018
§ Faculty of Business and Information Technology, University of Ontario Institute of Technology, Oshawa, ON, Canada
Email: {huangcheng, rxlu}@ntu.edu.sg, zhuhui@xidian.edu.cn, chn.junshao@gmail.com,
{Abdulrahman.Alamer, xiaodong.lin}@uoit.ca

***Abstract*—With the ubiquity of mobile devices, location-based social networking applications have been widely used in people's daily life. However, due to the importance and sensitivity of location information, these applications may lead to serious security issues for user's location privacy. To handle these location privacy challenges, in this paper, we propose an efficient and privacy-preserving proximity testing scheme, called EPPD, for location-based services. With EPPD, a group of users can test whether they are within a given distance with minimal privacy disclosure. In specific, EPPD is comprised of two phases: first, users periodically upload their encrypted locations to service provider; and later, users can send requests to service provider for proximity testing and obtain the final testing results. Detailed security analysis shows that EPPD can achieve privacy-preserving proximity testing. In addition, performance evaluations via extensive simulations also demonstrate the efficiency and effectiveness of EPPD in term of low computational cost and communication overhead.**

***Index Terms*—Location privacy, Proximity Testing, Privacy-preserving, Differential privacy**

## I. Introduction

In our society, as smartphones, tablets and other wearable devices spread, social networking applications, such as WhatsApp, Facebook, etc., have become significant and indispensable parts for people's daily life, e.g., the active user number of WhatsApp has reached 900 million by September 2015 and the user number is still proliferating [1]. Indeed, the convenience of smartphones and various functions of these applications can provide people with an even more intelligent and easier life in many aspects. Particularly, location-based services are the core functionality of social networking applications, and can offer plenty of useful services based on user's actual location, such as finding nearby friends, hailing a taxi and making a dating appointment.

Although the motivation of location-based social networking applications is to help people, most of these applications ignore the importance of privacy when tracking user's sensitive location, and may easily leak users' daily movements to some unintended or intended audiences. For instance, in July 2015, through an application called flightradar24, anyone can track the flight path of Prince William, which is extraordinary dangerous for protecting the future King from terror attack [2]. In fact, many location-based services today, provided by either the carriers or some companies, require users to upload their real locations without any privacy protection. While for convenience, users tend to acquiescently trust these companies, even though they clearly know that their locations could be leaked. Therefore, how to cope with the privacy challenges of location-based services has become one of the important research topics recently [3–9].

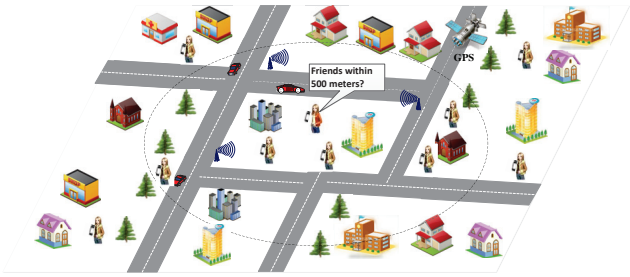



Fig. 1. An example of proximity testing

Motivated by the above-mentioned, in this paper, we aim to address the problem of *privacy-preserving proximity testing* in the context of location-based services. With privacy-preserving proximity testing [3], as shown in Fig. 1, friends can be notified when they are within a given distance of each other, e.g. 500 meters, but otherwise reveal no information about their exact locations to anyone. Although several privacy-preserving proximity testing schemes have been studied [3–9], most of them focus on private proximity testing between two parties. When deploying these schemes in real world, the efficiency and utility become some big issues with multiple parties. Different from those previously reported ones, based on a fast scale product technique [10] and differential privacy techniques [11, 12], we propose a novel privacy-preserving proximity testing scheme, called EPPD, which not only enables two parties to achieve privacy-preserving proximity testing, but also allows multiple parties to achieve privacy-preserving proximity testing efficiently at the same time. Specifically, the main contributions of this paper are three-fold.

- Firstly, we present an efficient and privacy-preserving proximity testing scheme, called EPPD. Unlike most existing

works, EPPD can support multiple users to collaboratively achieve proximity testing without exposing any location privacy of users or the testing results, i.e, user's location data and the testing results cannot be disclosed by any adversary.

• Secondly, through differential privacy techniques, EPPD can achieve a good trade-off of efficiency, accuracy and privacy when performing the privacy-preserving proximity testing.

• Finally, our proposed scheme is further implemented in Java. Compared with [6], we run extensive experiments to validate our proposed scheme in terms of computational cost and communication overhead. The performance analyses show that EPPD is effective with low computational cost and communication overhead. Meanwhile, the results of proximity testing are also accurate.

The remainder of this paper is organized as follows. In Section II, we introduce the system model, security model and design goal. Then, we propose an efficient and privacy-preserving proximity testing scheme in Section III. Subsequently, the security analysis and performance evaluation are given in Section IV and Section V, respectively. Finally, Section VI reviews some related works and Section VII draws some conclusions.

## II. Models and Design Goal

In this section, we formalize our system model, security model, and identify our design goal on privacy-preserving proximity testing.

### A. System Model

In our system model, we mainly consider three types of entities, namely a trusted authority (TA), a service provider (SP) and a larger number of registered users $U = \{U_1, U_2, ..., U_n\}$, as shown in Fig. 2.

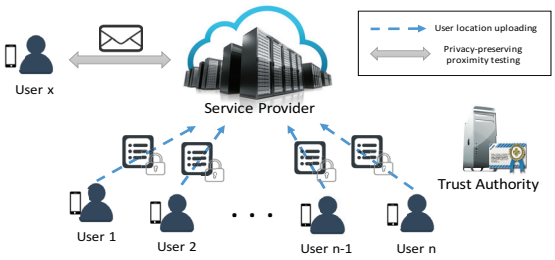



Fig. 2. System model under consideration

*Trusted authority* (TA): TA is a fully trustable and powerful entity, which is responsible for initializing the whole system and distributing key materials to others.

*Service Provider* (SP): SP is a core entity. The main function of SP is to store users' encrypted locations, to perform privacy-preserving proximity testing according to users' requests, and to relay the testing results to the corresponding user finally.

*Users* ($U = \{U_1, U_2, ..., U_n\}$): Each registered user $U_i \in U$ is equipped with smartphones and can obtain his/her current location. Users need to frequently upload their encrypted location data to SP for proximity testing.

In our model, TA first initializes the whole system. Then, there are two phases. The first phase is *user location uploading*. At every time interval, users upload their current encrypted location data to SP and SP stores them in database. The second phase is *privacy-preserving proximity testing*. When a user initiates a request for privacy-preserving proximity testing, SP performs the testing among multiple users to get the results, and relays the final results to the requesting user. Different from previous models [6], EPPD allows one user to achieve proximity testing among multiple users efficiently.

### B. Security Model

In our system, SP is *honest-but-curious*, which means he may faithfully follow the procedures of privacy-preserving proximity testing, but may be curious and try to disclose users' data when the condition is satisfied. Meanwhile, the adversary may try to compromise users' data by eavesdropping the communication package between users and SP. Thus, the following security requirements should be satisfied to guarantee the privacy of users' data in our model.

• *Privacy*: Users can upload their encrypted location data to SP and achieve privacy-preserving proximity testing without revealing any actual locations and testing results to the adversary. Notice that, similar as [3, 8], to improve the efficiency, we do not consider the collusion attack on privacy in our model, i.e., there is no collusion between users and SP.

• *Authentication and data integrity*: The data transmission should be confidential and authenticated that they are really generated by the corresponding entities, i.e. data cannot be disclosed, forged, modified and/or replayed by any adversary.

### C. Design Goal

Considering the aforementioned system model and security model, our design goal is to propose an efficient and privacy-preserving proximity testing scheme. Specifically, the following design goals should be satisfied: i) The security requirements should be guaranteed; ii) the proximity testing should be feasible and support multiple users; and iii) computational cost and communication overhead should be acceptable.

## III. Proposed EPPD Scheme

In this section, based on a fast scale product technique in [10] and differential privacy techniques [11, 12], we propose our EPPD scheme, which is mainly comprised of three parts: system initialization, user location uploading and privacy-preserving proximity testing. Before plunging into the details, we first introduce differential privacy [11] and geo-indistinguishability [12], which serve as the basis of EPPD.

### A. Differential Privacy and Geo-Indistinguishability

Differential privacy was first proposed by Dwork in 2006. By adding appropriately chosen noises (e.g. Laplace distribution) to the one dataset. The datasets, even having similar records, will become indistinguishable. Geo-indistinguishability, proposed by Andres et al., is a variation of

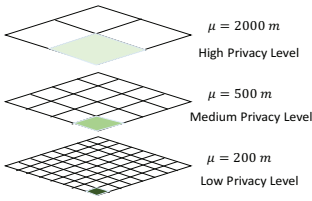
differential privacy under Euclidean metric which can provide location privacy protection. The formal definition of geo-indistinguishability is as follows:

***Definition 1 (Geo-Indistinguishability)*** A mechanism $K$ satisfies $\varepsilon$-geo-indistinguishability iff for all two different points $x$, $x'$:

$$K(x)(Z) \leq e^{\varepsilon d(x,x')} K(x')(Z).$$

Equivalently, the definition can be rewritten as $\frac{P(Z|x)}{P(Z|x')} \leq e^{\varepsilon d(x,x')}$, where $P$ is the conditional probability. Each observation is $Z \subseteq \mathcal{Z}$, where $\mathcal{Z}$ is a set of possible reported locations, and $d(x,x')$ is the Euclidean distance between $x$ and $x'$. By adding a planar laplacian noise $\mathcal{N}$, the real actual location can be reported as a cloaking location. Given the parameter $\varepsilon \in \mathbb{R}^+$, and the actual location $x_0 \in \mathbb{R}^2$, the probability density function of noise mechanism, on any other point $x \in \mathbb{R}^2$, is $D_\varepsilon(x_0)(x) = \frac{\varepsilon^2}{2\pi} e^{-\varepsilon d(x_0,x)}$, which can also be represented as polar coordinate model $D_\varepsilon(r,\theta) = \frac{\varepsilon^2}{2\pi} r e^{-\varepsilon r}$, where $r$ and $\theta$ are distance and angle with respect to $x_0$. To cloak the real location, specifically, $\theta$ should be uniformly chosen from $[0, 2\pi)$ and $r$ should be set as $C_\varepsilon^{-1}(p) = -\frac{1}{\varepsilon}(W_{-1}(\frac{p-1}{e})+1)$, where $W^{-1}$ is the Lambert $W$ function (the -1 branch) and $p$ should be uniformly chosen from $[0, 1)$. We refer to [12] for more detailed descriptions of geo-indistinguishability.

### B. System Initialization




(a) Grid areas

| | | | | |
|---|---|---|---|---|
| 0,0 | 0,1 | 0,2 | 0,3 | 0,4 |
| 1,0 | 1,1 | 1,2 | 1,3 | 1,4 |
| 2,0 | 2,1 | 2,2 | 2,3 | 2,4 |
| 3,0 | 3,1 | 3,2 | 3,3 | 3,4 |
| 4,0 | 4,1 | 4,2 | 4,3 | 4,4 |

(b) Area tags

Fig. 3. Grid areas and area tags

A single trusted authority (TA) takes the charge of bootstrapping the whole system and generating public and private keys for registered users $U$ and service provide (SP). Specifically, TA executes the bootstrap as follows:

- Given the security parameters $k_1$, $k_2$ and $k_3$, TA chooses two large primes $p$ and $\alpha$, such that $|p| = k_1$ and $|\alpha| = k_2$, and chooses a large random number $s \in \mathbb{Z}_p^*$ as the secret key; then, TA generates a cyclic group $G$ of prime order $p$, in which the discrete logarithm problem (DLP) is hard, and $h \in G$ is a random generator of $G$.
- TA defines one public cryptographic hash functions $H : \{0,1\}^* \rightarrow G$.
- For each registered users $U_i \in U = \{U_1, U_2, ..., U_n\}$, with the different identity $ID_i$, TA chooses a random number $s_i \in \mathbb{Z}_p^*$ as $U_i$'s private key, and computes $S_i = h^{s_i}$ as $U_i$'s public key; then, TA assigns key materials to the corresponding users; finally, TA shares secret key $s$ with these registered users.
- TA generates identity $ID_c$ for SP, chooses private key as $s_c \in \mathbb{Z}_p^*$ and computes the public key as $S_c = h^{s_c}$; then, TA assigns them to SP.
- Given the parameter $\varepsilon \in \mathbb{R}^+$, TA divides the whole map into many grid areas of which the side length $\mu$ are same, as shown in Fig. 3 (a). Each area has a area tag $tag_{i,j}$, e.g., $tag_{i,j} = (i,j) = (1,2)$, and each registered user's location can be mapped into one of these areas in the map, such as $(x,y) \rightarrow tag_{i,j}$, as shown in Fig. 3 (b).
- Finally, TA publishes area information and public parameters as $< G, h, p, \alpha, ID_c, S_c, H, \mu, \varepsilon, k_3 >$. For each registered user $U_i \in U$, TA publishes $< ID_i, S_i >$ as system-wide public information.

Meanwhile, the advanced encryption standard (AES) is adopted as the symmetric cryptosystem in our system. Denote $AES\text{-}ENC_k$ and $AES\text{-}DEC_k$ as the encryption and decryption algorithms under the symmetric key $k$, respectively. Note that, the side length $\mu$ of area, decided by TA, can determine the privacy level of users' locations according to geo-indistinguishability. When $\mu$ is different (e.g. 200 meters, 500 meters and 2000 meters), the privacy level of registered users' locations are different, such that the longer $\mu$ is, the higher privacy level can be achieved.

### C. User Location Uploading

Based on GPS system, users can always obtain their latitude and longitude coordinates in smartphone but cannot directly gain xy-coordinate in Cartesian plane. In order to conveniently compute Euclidean distance between two coordinates, users need to convert the GPS data to xy-coordinate by Miller Projection. In our scheme, at every time interval, e.g., every 5 minutes, each user $U_i \in U$ uploads his/her current location data to SP, and SP stores each user's encrypted location data in database for possible private proximity testing subsequently.

*Step-1* At time point $t_\gamma$, each user $U_i$ can encrypt his/her current location $m = (x_i, y_i)$, where $x_i$ and $y_i$ mean $x$ and $y$ coordinate in Cartesian plane, and reports these location data to SP as follows:

• According to $U_i$'s current coordinates $m$, $U_i$ computes the cloaking location $m' = (x_i', y_i')$ as $x_i' = x_i + r\cos\theta$ and $y_i' = y_i + r\sin\theta$, where $r = -\frac{1}{\varepsilon}(W_{-1}(\frac{p-1}{e})+1)$, $p \in [0,1)$ and $\theta \in [0, 2\pi)$; then, $U_i$ can map his/her cloaking location $m'$ to an area tag $P_i$ based on $m'$ and area information.

• $U_i$ chooses two random number $r_{i1}$ and $r_{i2}$, whose length is $k_3$, and then encrypts the coordinate data $m = (x_i, y_i)$ as $C_i = (cx_i, cy_i)$.

$$cx_i = s \cdot (\alpha x_i + r_{i1}) \bmod p \tag{1}$$

$$cy_i = s \cdot (\alpha y_i + r_{i2}) \bmod p \tag{2}$$

• $U_i$ computes a non-interactive session symmetric key $k_{ic}$ shared with SP as $k_{ic} = H(S_c^{s_i} || ID_c || ID_i || H(t_\gamma))$ and then uses $k_{ic}$ to perform AES encryption as $C_i' = AES\text{-}ENC_{k_{ic}}(C_i || P_i || ID_c || ID_i || t_\gamma)$. Finally, $U_i$ sends the data $< C_i', ID_i, t_\gamma >$ to SP.

*Step-2* After receiving all uploaded data from each user $U_i \in U$, SP processes the encrypted data and store them in database as follows:

• SP checks the validity of the time interval between current time $t'_\gamma$ and $t_\gamma$ in order to resist the replaying attack. If $|t'_\gamma - t_\gamma| \leq \Delta T$, where $\Delta T$ denotes the expected valid time interval for transmission delay, SP accepts and processes $< C'_i, ID_i, t_\gamma >$, and rejects otherwise.

• For each user's reporting data $< C'_i, ID_i, t_\gamma >$, SP computes the corresponding non-interactive session key $k_{ci}$ as $k_{ci} = H(S_i^{s_c}||ID_c||ID_i||H(t_\gamma)) = k_{ic}$ and decrypts $C'_i$ as $AES\text{-}DEC_{k_{ci}}(C'_i) = C_i||P_i||ID_c||ID_i||t_\gamma$. Finally, SP stores $< C_i, P_i, ID_i, t_\gamma >$ in database.

### D. Privacy-preserving Proximity Test

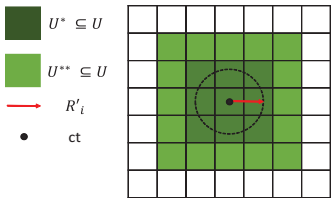



Fig. 4. Filter users with area tags

In order to achieve privacy-preserving proximity testing, users and SP need to perform the following steps.

*Step-1* Assume that $U_i \in U$ wants to perform privacy-preserving proximity testing at time point $t_q$, he/she need to set a testing range $R_i$ and reports to SP as follows:

• $U_i$ chooses a uniformly distributed random number $\sigma \in [0, 1)$, and then computes the cloaking testing range $R'_i$ as $R'_i = R_i - \frac{1}{\varepsilon}(W_{-1}(\frac{\sigma-1}{e}) + 1)$.

• $U_i$ chooses two random number $r'_{i1}$ and $r'_{i2}$, whose length is $k_3$, and then computes $Rq_1 = s \cdot (\alpha R_i + r'_{i1}) \bmod p$ and $Rq_2 = s^2 \cdot \alpha^2 \cdot r'_{i2} \bmod p$.

• $U_i$ generates a non-interactive session symmetric key $k_{ic} = H(S_c^{s_i}||ID_c||ID_i||H(t_q))$ to encrypt $R'_i$, $Rq_1$ and $Rq_2$ as $Rq' = AES\text{-}ENC_{k_{ic}}(Rq_1||Rq_2||R'_i)$. Finally, $U_i$ sends the testing request $< Rq', ID_i, t_q >$ to SP.

*Step-2* SP receives $U_i$'s testing request and performs privacy-preserving proximity testing as follows:

• SP first checks the time stamp $t_q$ and then computes $k_{ci} = H(S_i^{s_c}||ID_c||ID_i||H(t_q))$ to decrypt $Rq'$ as $AES\text{-}DEC_{k_{ci}}(Rq_1||Rq_2||R'_i)$.

• According to $U_i$'s area tag $P_i$ stored in database, SP can find the centre of $P_i$, denoted as $ct$.

• As shown in Fig. 4, SP starts from $ct$ as centre, circles with radius equal to $R'_i$ and gets a circle region $O_i$. According to each user $U_x$'s area tag $P_x$, where $x \in [1, n]$ and $x \neq i$, SP locates the users $U^*$ whose areas intersect with $O_i$ and the users $U^{**}$ whose areas are neighbor to $U^*$'s areas. For example, area $(i, j)$'s neighbors includes $(i, j)$, $(i-1, j-1)$, $(i-1, j)$, $(i-1, j+1)$, $(i, j-1)$, $(i, j+1)$, $(i+1, j-1)$ and $(i+1, j+1)$. Then, SP combines $U^*$ and $U^{**}$ as $U' = \{U^*, U^{**}\}$.

• For each user $U'_x \in U'$, SP chooses a random number $r_{qix}$, whose length is $k_3$, and computes $T_{ix} = r_{qix} \cdot (Rq_1^2 - (cx_i + cx_x)^2 - (cy_i + cy_x)^2) + Rq_2 \bmod p$. Next, SP sends the query $T_{ix}$ to $U'_x$.

• After receiving the query, $U'_x$ computes $Q_{ix} = s^{-2} \cdot T_{ix}$. Based on the bit length of $Q_{ix}$, $U'_x$ can determine the results $A_{ix}$ (True/False) of private proximity testing. If the length of $Q_{ix}$ is close to $k_1$, i.e., $|Q_{ix}| \approx k_1$, $U'_x$ can judge he/she is not in the test range of $U_i$, which means $A_{ix} = False$, otherwise, he/she is in the testing range, which means $A_{ix} = True$. At the time point $t_{\beta x}$, $U'_x$ computes the non-interactive session symmetric key $k_{ix}$ as $k_{ix} = H(S_i^{s_x}||ID_i||ID_x||H(t_{\beta x}))$ and encrypts $A_{ix}$ as $A'_{ix} = AES\text{-}ENC_{k_{ix}}(A_{ix}||ID_i||ID_x||t_{\beta x})$. Subsequently, $U'_x$ sends the response $< A'_{ix}, ID_x, t_{\beta x} >$ back to SP.

• SP receives all responses and relays all responses $A = \{< A'_{i1}, ID_1, t_{\beta 1} >, < A'_{i2}, ID_2, t_{\beta 2} >, ..., < A'_{in}, ID_n, t_{\beta n} >\}$ together to $U_i$.

*Step-3* After receiving $A$, for each user $U'_x$'s response $< A'_{ix}, ID_x, t_{\beta x} >$, $U_i$ computes non-interactive session key $k_{xi}$ as $k_{xi} = H(S_x^{s_i}||ID_i||ID_x||H(t_{\beta x})) = k_{ix}$ and decrypts $A'_{ix}$ as $AES\text{-}ENC_{k_{xi}}(A_{ix})$. Finally, $U_i$ obtains the testing result $A_{ix}$ for each user $U'_x \in U'$.

## IV. Security Analysis

In this section, we analyze the security properties of the proposed scheme. In specific, our analyses focus on how the proposed scheme achieves all the security requirements defined earlier.

• *Users' location and the proximity testing is privacy-preserving* i) User $U_i$ uploads the encrypted location $C_i = (cx_i, cy_i)$. $C_i$ is onetime masked with random $cx_i = (\alpha x_i + r_{i1}) \bmod p$ and $cy_i = s \cdot (\alpha y_i + r_{i2}) \bmod p$. $s$ is the secret key sharing among users. $r_{i1}$ and $r_{i2}$ are two random number to ensure that every $C_i$ is different even though $x_i$ and $y_i$ are same. ii) User $U_i$ uploads the area tag $P_i$ to SP. According to geo-indistinguishability, $U_i$ gets $x'_i$ and $y'_i$ by adding the planar laplacian noise to $x_i$ and $y_i$. In this manner, $\frac{P(Z|x)}{P(Z|x')} \leq e^{\varepsilon d(x,x')}$ has been guaranteed, which means differential privacy of user's location has been achieved. iii) User $U_i$ sends the requests $Rq_1 = s \cdot (\alpha R_i + r'_{i1}) \bmod p$, $Rq_2 = s^2 \cdot \alpha^2 \cdot r'_{i2} \bmod p$ and $R'_i$ to SP. Without the secret key $s$, $Rq_1$ and $Rq_2$ are privacy-preserving for SP and the $R'_i$ is changed by planar laplacian noise. Therefore, users' actual location and the proximity testing is privacy-preserving.

• *The authentication and data integrity have been achieved* i) when performing user location uploading, $U_i$ computes a non-interactive session key $k_{ic} = H(S_c^{s_i}||ID_c||ID_i||H(t_\gamma)) = H(h^{s_c s_i}||ID_c||ID_i||H(t_\gamma))$ and encrypts his/her data by $AES\text{-}ENC_{k_{ic}}(data)$. SP can authenticate the data by computing $k_{ci} = H(S_i^{s_c}||ID_c||ID_i||H(t_\gamma)) = H(h^{s_c s_i}||ID_c||ID_i||H(t_\gamma)) = k_{ci}$ and decrypts the data as $AES\text{-}DEC_{k_{ci}}(data)$. $t_\gamma$ is the current time for preventing replaying attack. With SP's public key and $U_i$'s privacy key $s_i$, $S_c^{s_i}$ is a signature of $U_i$. SP can verify the signature with $U_i$'s public key $S_i$ and his

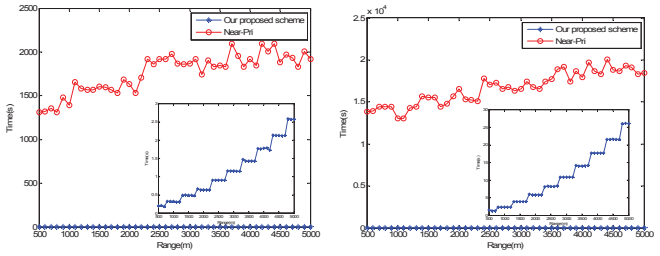
private key $s_c$ as $S_i^{s_c}$. ii) When $U_x$ sends the testing result back to $U_i$, he/she also computes the session key $k_{ix} = H(S_i^{s_x}||ID_i||ID_j||H(t_{\beta x})) = H(h^{s_i s_x}||ID_i||ID_j||H(t_{\beta x}))$. By performing the AES encryption with $k_{ix}$, the testing result has been encrypted and only $U_i$ can decrypt the data with $k_{xi} = H(S_x^{s_i}||ID_i||ID_j||H(t_{\beta x}))$ and get the final testing result. Therefore, the authentication and data integrity have been achieved.

## V. Performance Evaluation

In this section, we evaluate the performance of EPPD in terms of *computational cost*, *communication overhead*, and *accuracy rate of testing results*. We compare our proposed scheme with the state-of-the-art privacy-preserving proximity testing scheme, called Near-pri [6]. Near-pri is mainly based on homomorphic encryption techniques and uses binary tree to reduce computational cost. We extend the privacy-preserving proximity two-party testing scheme to support proximity testing among multiple parties as follows.

If user A wants to perform testing, he/she first encrypts current location using his/her public key and sends the requests to SP; then, SP relays the requests to all users except user A; Other users encrypt their locations using user A's public key and upload the data to SP; after receiving all responses, according to homomorphic encryption techniques, SP helps user A to achieve testing and sends results back to user A; finally user A decrypts the results using his/her private key and gets the final testing results.

In our simulations, total $l$ users $U = \{U_1, U_2, ..., U_l\}$ are first uniformly deployed in a map of $40\ km\ *\ 40\ km = 1600\ km^2$. The detailed parameters are set as follows: $\epsilon = 2$, $\mu = 500$, $k_1 = 1024$, $k_2 = 200$ and $k_3 = 128$. Our experiment environment is a laptop with 3.1 GHz processor, 8GB RAM, and Window 7 platform. In addition, we also implement the scheme of EPPD and Near-pri in Java.




(a) $l = 10,000$ (b) $l = 100,000$

Fig. 5. Computational cost of privacy-preserving proximity testing

### A. Computational cost

Considering the computational cost of user location uploading, we run the phase 10000 times to get the average computation time $t \approx 10$ ms, which is clearly very efficient.

When performing privacy-preserving proximity testing, the computational cost is primarily dependant on two factors. One is the population density in the testing map and the other is the size of testing range. The total testing area is fixed as $1600\ km^2$, so we change different user number $l = 10,000$ and $l = 100,000$ and set various testing range $R$, from small (500 meters) to larger (5000 meters) in our simulation, as shown in Fig. 5 (a) and (b).

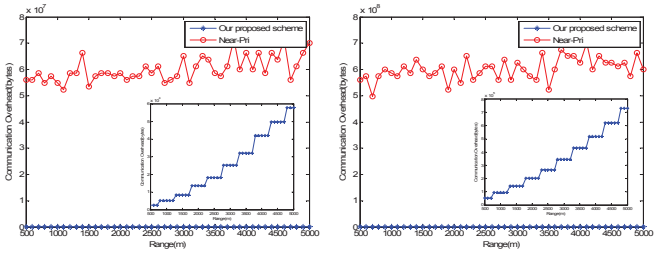
Obviously, without exponentiation operation in Paillier encryption, our scheme outperforms the scheme of Near-pri in computational cost and supports multiple users to achieve privacy-preserving proximity testing. The running time of Near-pri is above 1000 s when user number is 10,000. However, the running time of EPPD is almost 300 ms to 3000 ms with the increase of $R$. It is clear that Near-pri need to test every user no matter what the testing range is, therefore, the cost is stable with the increase of $R$.

Here, our simulation runs in one single thread. Thus, when user number increases to 100,000, the computational cost also increases. In consideration of deploying the whole system in real world, SP can perform the privacy-preserving proximity testing in multiple threads and achieve computation concurrently to enhance the computational efficiency.

### B. Communication overhead

When users upload their encrypted locations, the communication overhead is 292 bytes, which is acceptable for smartphones. For user $U_i$, the uploaded data is $< C_i', ID_i, t_\gamma >$, where $C_i' = AES\text{-}ENC_k(C_i||P_i||ID_i||ID_j||t_\gamma)$. According to our scheme, the byte length of each data is $|C_i| = 256$, $|ID_i| = |ID_j| = 4$ and $|P_i| = |t_\gamma| = 8$.

When performing the privacy-preserving proximity testing, similar as above, the communication overhead will increase with the increase of testing range and population density, as shown in Fig. 6 (a) and (b).




(a) $l = 10,000$ (b) $l = 100,000$

Fig. 6. Communication overhead of privacy-preserving proximity testing

Compared with Near-pri, the communication overhead of our scheme is lower. Through differential privacy techniques, we can filter and eliminate most of users and leave a few users to perform privacy-preserving proximity testing, which can effectively reduce the communication times in the whole system. Specifically, when user number is 10,000, the communication overhead is almost 3000 bytes to 60,000 bytes with the increase of testing range $R$. When user number is 100,000, the communication overhead is almost 50,000 bytes to 740,000 bytes for each testing. However, the communication overhead of Near-pri is almost 60,000,000 bytes, which is far more than EPPD's communication overhead.

### C. Accuracy rate of testing results

TABLE I
ACCURACY RATE OF TESTING RESULTS

| $\sigma$ | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 |
|---|---|---|---|---|---|
| Accuracy | 97.7% | 97.8% | 98.3% | 98.5% | 99.0% |
| $\sigma$ | 0.6 | 0.7 | 0.8 | 0.9 | 0.99 |
| Accuracy | 99.1% | 99.2% | 99.3% | 99.3% | 99.5% |

Due to differential privacy and noises added to registered users' locations, the final testing results may miss some users who are in testing range. According to geo-indistinguishability, the accuracy of testing results is controlled by users' setting $\sigma$. In order to analyze the effects of $\sigma$ on testing results, we set $l = 10000$, $R = 2500$ and choose different $\sigma$ from $0.1$ to $0.99$. By running EPPD 1000 times, we evaluate the average accuracy rate as shown in Table. I. The accuracy rate of testing results is almost 99% with different $\sigma$, which means the final testing results are almost accurate with minimum error.

## VI. Related Work

Recently, there are several privacy-preserving proximity testing schemes proposed for location-based services, we briefly review some of them [3–9] in this section.

*Privacy*, *Functionality* and *Efficiency* are three important factors of private proximity testing. To achieve these factors, based on private set intersection (PSI), Narayanan et al. [3] first introduced a private proximity testing scheme by private equality testing (PET) with location tags. To improve privacy and secure level, Nielsen et al. [4] proposed a active secure scheme which can handle more possible attacks in private proximity testing, and Saldamli et al. [5] proposed a scheme supporting testing through a untrusted server. Novak and Li [6] uses Paillier encryption to propose a new private proximity testing scheme by making use of binary tree techniques to reduce computational cost. Also, Kotzanikolaou et al. [7] proposed a more efficient scheme based on PET, but their scheme does not support setting testing range for users. For a more general scheme, Sedenka and Gasti [8] proposed a private proximity testing scheme supporting different coordinate systems on earth based on homomorphic encryption. Besides, without third-party server, Yao Zheng et al. [9] proposed a scheme which does not only support location based handshake but also supports private proximity testing with location tags.

Different from the above schemes which are designed for private proximity testing between two parties, our proposed scheme supports private proximity testing among multiple users, thus EPPD can be more practical and efficient in real world.

## VII. Conclusion

In this paper, we have proposed an efficient scheme for privacy-preserving proximity testing, called EPPD. With multiple users and their encrypted locations, the service provider can achieve proximity testing without privacy leakage. Detailed security analysis shows that EPPD is privacy-preserving, i.e., no one can disclose other users' location data, and only the corresponding user, who sends the request, can decrypt the final testing results. In addition, through extensive performance evaluation, we have also demonstrated that EPPD is efficient in terms of computational costs and have low communication overhead. In future work, we may take collusion attack into consideration and achieve a more secure scheme for proximity testing.

## Availability

The Java implementation of the proposed EPPD can be downloaded at http://www.ntu.edu.sg/home/rxlu/project/index.htm.

## References


[1] L. Rao, "Whatsapp hits 900 million users," http://fortune.com/2015/09/04/whatsapp-900-million-users/, 2015, [Online; accessed 09-October-2014].

[2] P. Joshi, "Prince william: Security alert over mobile phone app that tracks duke of cambridge's air ambulance helicopter," http://www.ibtimes.co.uk/prince-william-security-alert-over-mobile-phone-app-that-tracks-duke-cambridges-air-ambulance-1512573, 2015, [Online; accessed 03-September-2015].

[3] A. Narayanan, N. Thiagarajan, M. Lakhani, M. Hamburg, and D. Boneh, "Location privacy via private proximity testing," in *NDSS 2011*, 2011.

[4] J. D. Nielsen, J. I. Pagter, and M. B. Stausholm, "Location privacy via actively secure private proximity testing," in *PerCom 2012*, 2012, pp. 381–386.

[5] G. Saldamli, R. Chow, H. Jin, and B. P. Knijnenburg, "Private proximity testing with an untrusted server," in *WISEC'13*, 2013, pp. 113–118.

[6] E. Novak and Q. Li, "Near-pri: Private, proximity based location sharing," in *INFOCOM 2014*, 2014, pp. 37–45.

[7] P. Kotzanikolaou, C. Patsakis, E. Magkos, and M. Korakakis, "Lightweight private proximity testing for geospatial social networks," *Computer Communications*, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0140366415002558

[8] J. Sedenka and P. Gasti, "Privacy-preserving distance computation and proximity testing on earth, done right," in *ASIA CCS '14*, 2014, pp. 99–110.

[9] Y. Zheng, M. Li, W. Lou, and Y. T. Hou, "SHARP: private proximity test and secure handshake with cheat-proof location tags," in *ESORICS 2012*, 2012, pp. 361–378.

[10] R. Lu, H. Zhu, X. Liu, J. K. Liu, and J. Shao, "Toward efficient and privacy-preserving computing in big data era," *IEEE Network*, vol. 28, no. 4, pp. 46–50, 2014.

[11] C. Dwork, "Differential privacy," in *ICALP 2006*, 2006, pp. 1–12.

[12] M. E. Andrés, N. E. Bordenabe, K. Chatzikokolakis, and C. Palamidessi, "Geo-indistinguishability: differential privacy for location-based systems," in *CCS'13*, 2013, pp. 901–914.