This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2015.2472529, IEEE Transactions on Dependable and Secure Computing



# Location Based Handshake and Private Proximity Test with Location Tags

Yao Zheng, Ming Li, Wenjing Lou, and Y. Thomas Hou

**Abstract**—A location proximity test service allows mobile users to determine whether they are in close proximity to each other, and has found numerous applications in mobile social networks. Unfortunately, existing solutions usually reveal much of users' private location information during a proximity test. They are also vulnerable to location cheating where an attacker reports false locations to gain an advantage. Moreover, the initial trust establishment among unfamiliar users in large scale mobile social networks has been a challenging task. In this paper, we propose a novel scheme that enables a user to perform (1) a location based handshake that establishes secure communications among strangers, who do not have a pre-shared secret, and (2) a privacy-preserving proximity test without revealing the user's actual location to the server or other users not within the proximity. The proposed scheme is based on a novel concept, *i.e.*, spatial-temporal location tags, and we put forward a location tag construction method using environmental signals that provides an unforgeable location proof. We use Bloom filters to efficiently represent users' location tags and vicinity regions. We exploit fuzzy extractor, a lightweight cryptographic primitive, to extract shared secrets between matching location tags. We conduct extensive analysis, simulation, and real experiments to demonstrate the feasibility, security, and efficiency of our scheme.

**Index Terms**—location-based service, proximity test, spatial-temporal location tag, location cheating, location privacy, fuzzy extractor, Bloom filter.

## 1 Introduction

The proliferation of smartphones has given rise to location-based service (LBS), which has drawn considerable research attention in recent years. The key enabler of LBS is the availability of users' locations, which can be easily measured and reported by smartphones today. With LBS, users report their locations in real-time to a location server, which allows users to ubiquitously query places of interest around them, or test if their friends are within certain physical proximity. The latter is called "*proximity test*" [1] and has found numerous mobile applications, for example, to locate nearby friends (*e.g.*, in a mobile social network [2]), in an emergency situation to find nearby medical personnel (*e.g.*, in mobile healthcare [3], [4]), or to search for nearby Ubers, in mobile transportation services, only to name a few. The former is representative of proximity test between friends, while the latter are examples of proximity test between strangers, who may not share any secrets *a priori*.

Similar to many LBSs, there are many security and privacy concerns associated with proximity test that may hinder its widespread adoption. One of the concerns is that the reported locations could be easily forged by malicious users in order to exploit the benefits of proximity test services. There are many incentives for users to not report their locations truthfully. For example, in [5] a *location cheating* attack has been discovered in which the attacker reports false locations to gain revenue by acquiring shopping coupons. In addition, a curious user may try to profile other users' locations by setting hers to any desired place. Thus, it is essential to provide an unforgeable location proof in proximity matching, so as to ensure the social welfare of LBSs. On the other hand, the *location privacy* is also an important concern for common users. The primary reason is that the location servers are often operated by third-party service providers such as cloud platforms, which tend to not be fully trusted by people [6]. Meanwhile, users may not want their friends or strangers in the system to know about their exact locations and track them down.

To design a privacy-preserving proximity test scheme that is also cheat-proof involves several challenges. First, given the mobile and distributed nature of LBS users, how can we make sure that a user's reported location is truthful without involving a trusted authority? Some researchers suggest a distributed proof approach using presence evidence from peer devices [7]. Although the idea is intriguing, the anonymization process involves using multiple public/private key pairs as pseudonyms, which would require significant modifications to the existing public-key infrastructure (PKI). Second, shared keys are usually required for preserving privacy during proximity test and securing communications between matched users. However, the initial trust establishment among users in a large-scale mobile social network remains a difficult task, simply because managing shared keys with everyone else is not scalable without a trusted authority. Most existing solutions to date have relied on *a priori* shared secrets between each pair of users [1], [8], which severely limits their applicability and scalability. Finally, efficiency and usability need to be achieved simultaneously. To achieve a strong privacy guarantee, previous protocols either rely on computationally intensive cryptographic primitives, or do so at a cost of high communication overhead.

In this paper, we propose a novel scheme that performs

*Manuscript received . . . ; revised . . . ; accepted . . . .*
*This research is based upon work supported by the National Science Foundation under Grant No. 1446478, 1443889, 1405747, 1217889, 1218085.*
*Y. Zheng and W. Lou are with the Department of Computer Science, Virginia Tech, USA e-mail: {zhengyao,wjlou}@vt.edu*
*Y. Thomas Hou is with the Department of Electrical and Computer Engineering, Virginia Tech, USA e-mail: thou@vt.edu*
*M. Li is with the Department of Electrical and Computer Engineering, University of Arizona, USA e-mail: lim@email.arizona.edu*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2015.2472529, IEEE Transactions on Dependable and Secure Computing



a location based handshake and private proximity test. We focus on a general one-to-many proximity match setting, which allows a user to find out from a group of users the one(s) that are within her vicinity region[1] with the help of a semi-trusted server. In order to defeat location cheating, we propose a novel form of location representation, the *spatial-temporal location tag*, which is constructed from radio signals captured in a device's surrounding environment, such as WiFi and LTE signals. An attacker cannot forge a location tag if she is not at the corresponding location and time, due to the high freshness (entropy) and spatial variety of environmental signals. We make use of the *Bloom filter* to efficiently represent users' location tags and vicinity region. We exploit the *fuzzy extractor* [10], a lightweight cryptographic primitive, to extract secret keys automatically between users based on their location tags, while ensuring that a user's location is revealed to neither the server nor users who are outside of the vicinity region.

The main contributions of this paper are as follows. (1) We propose a novel form of user location representation, *i.e.*, spatial-temporal location tag, to defeat location cheating attacks in LBSs. We demonstrate our concept using collected real-world WiFi and LTE signal traces and employ entropy analysis to show the feasibility of generating unforgeable location tags in practice. (2) We propose a novel location based handshake and private proximity test protocol based on spatial-temporal location tags, which establishes security associations and performs proximity test between one user and many others at the same time. We uniquely combine the Bloom filter and the fuzzy extractor to meet the stringent privacy and efficiency requirements. Our protocol avoids the complexity of key management among users as it does not rely on pre-shared secrets. (3) We carry out both thorough analysis and performance evaluation. We identify the security and privacy benefits for introducing the spatial-temporal location tag into proximity test. Then we study the protocol's functionality and efficiency using simulations and real-world experiments. We show that our protocol supersedes existing protocols for fine grained proximity test. To the best of our knowledge, this is the first work that systematically studies an unforgeable location tag and its use in LBSs.

## 2 Related Work

For privacy in LBSs, most previous works have been focusing on privacy in location queries, *i.e.*, a model in which users report their "encrypted" location data to a central database server to perform range or k-nearest-neighbor (kNN) queries [11], [12]. Note that in this model the database stored in the server is assumed to be public. In contrast, the recently emerged proximity test is a different model where location-based matching is done only between users, while the users' locations are private information. Here we briefly describe the concepts of proximity test and private matching.

**Proximity Test:** Proximity test is a special form of location sharing [6], where the information being shared is whether or not two users are within a certain range or in the same geographic region. The main privacy concern in proximity test is that user's actual location may be involuntarily revealed to either the server or other users. To this end, a privacy-preserving proximity test solution is proposed in [8], using a grid-based encryption algorithm. In [13], Rasmussen *et al.* devised a privacy-preserving distance bounding protocol, using stream cipher. However, their protocol relies on a pre-shared secret key to initialize, and is subject to dictionary attack [14], [15]. In [16], Mascetti *et al.* proposed proximity detection schemes based on service provider filtering, in which privacy protection is achieved by user-chosen location representation that controls its granularity. However, their protocol leaks coarse-grained location information to the server. In [1], Narayanan *et al.* proposed a suite of private proximity test protocols. The possibility of constructing location tags from environmental signals was noted; however, their protocols either require a pre-shared secret key between users, or is not scalable and efficient enough to handle one-to-many proximity test as studied in our paper. Another proximity test scheme was proposed in [9], where users can also control their privacy levels via leveled publishing. The protocol is based on keyed hashing which suffers from dictionary attacks. In [17], Lin *et al.* proposed a proximity test scheme by applying shingling technique [18] to GSM cellular messages. However, they did not thoroughly analyze its security. In our previous work [19], we designed a two-step private proximity test protocol using unforgeable location tags. However, the original protocol cannot detect fine-grained location cheating in the second step. In this paper, we provide an improved protocol that can be robust against fine-grained location cheating using an additional peer-to-peer location proof mechanism. We carry out a systematic study of unforgeable location tags, and their usage in proximity test based on a more thorough analysis, realistic simulations, and more comprehensive real-world experiments.

**Private Matching:** Our proposed scheme constructs location tags and takes the location tags as the inputs to a private matching scheme to realize proximity test. Different location tag construction methods will yield different types of location tags with different data structure representation, which in turn demands different secure matching algorithms. Secure inner product computation has been proposed to compute the number of matching keywords between two binary-valued vector inputs, where each bit in the vector represents the presence or absence of a keyword [20]. Secure multi-party computation (SMC) techniques have also been used in private matching. For example, in [21], Freeman *et al.* proposed a private set intersection protocol using homomorphic encryption, where the inputs to be matched are two sets of elements. In this paper, we are matching two location tags, which are environmental signals represented using bloom filter and further coded using BCH coding. The method used to realize the private matching is also very different from previously known private matching methods. Essentially, our matching method is based on polynomial reconstruction, which is more efficient compared to previous private matching algorithms.

1. In proximity test, the vicinity region must contain the user's current location [9].

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2015.2472529, IEEE Transactions on Dependable and Secure Computing



## 3 System Model, Threat Model, and Design Goals

Here we describe the system model, threat model, and design goals of our protocol. We consider a general one-to-many proximity test setting, in which a user wishes to find out from a group of candidates the one(s) that are within certain proximity with the help of a centralized server. We refer to the user who initiates the proximity test as Alice, and an arbitrary candidate as Bob. Additionally, we consider the scenario that Alice has never met the candidates. Therefore, their devices do not have any prior associations to form secure communication channels. Such a setting is common in LBSs. In one scenario, Alice, who is in an emergency situation, may wish to test her proximity with nearby emergency medical technicians (EMTs). In another scenario, Alice, who is at a shopping mall, may wish to discover her proximity with nearby people to trade particular types of coupons. Since Alice may have never met these people before, it is unlikely that her devices have any pre-shared secret with them.

For the threat model, we assume that Alice and a strict majority of candidates are honest about their location information. An adversary is a malicious candidate who tries to use fraudulent location information to answer a proximity test and gain knowledge about Alices location. For instance, stalking is an example of the threat from this type of adversary. We further assume an honest-but-curious server who follows the protocol faithfully but is curious about users location information.

We consider four design goals in terms of functionality, security, privacy, and efficiency, as listed below.

**Location Based Handshake:** The main motivation of our study is to address the situation when Alice wishes to test her proximity with a group of users she has not met. Hence, a handshake protocol shall be implemented first, which allows users to establish a secure channel for subsequent communication. The handshake protocol must be a one-way broadcast since it can be a daunting task for Alice to identify the qualified users and establish security association individually.

**Security:** The main security goal for proximity test is to design an unforgeable location proof so that the protocol is robust against location cheating. Location cheating happens when one party is able to deceive the other party with an untruthful location. In our case, if Bob can trick Alice into believing that he is within her vicinity region while he actually is not, he has successfully launched a location cheating attack. Unforgeable location proofs are extremely important for location based services. To the best of our knowledge, we are among the first to address location unforgeability in proximity test.

**Privacy:** The privacy goal of the protocol is to maintain each users' location privacy. Specifically, through the proximity test, the server can not learn any users' locations. Alice should only learn the identities of the candidates who are within her vicinity region but not the exact locations of those candidate. Alice should not learn anything about the candidates who are outside of her vicinity region. The candidates should not learn whether they pass Alice's proximity test or anything that reveals Alice's location or the location of her vicinity region.

**Efficiency:** Existing private proximity test protocols [1], [8] operate on pairs of users. If Alice wants to test a group of $N$ users, she has to run the protocol $N$ times with every user in the group. This results in a bandwidth complexity of $\mathcal{O}(N)$ and a computational complexity of $\mathcal{O}(N)$ at Alice's side. Our goal is to design an efficient protocol where Alice and each participant only submit their information once to the server. This leads to a communication complexity of $\mathcal{O}(1)$ at the user's side. This represents a significant efficiency improvement compared to the existing schemes.

## 4 Location Tags from Environmental Signals

Introduced in [1], [22], [23], a location tag can be regarded as a token of proof associated with a point in space and time. It is collections of signals presented at a certain location at a certain time. Formally, we can define a location tag as follows:

***Definition 1.*** Let $\mathcal{X}$ be a set of environmental signals captured at a point in space and time. A location tag $\mathcal{Y}$ is a subset of $\mathcal{X}$, selected by a filter function $\phi\colon \mathcal{X} \rightarrow \mathcal{Y} \subseteq \mathcal{X}$. Each element, $y$, of $\mathcal{Y}$ is an observation of the location tag.

From the functionality point of view, a good location tag should at least have the *reproducibility* property. That is, if two measurements at the same space and time yield tags $\mathcal{Y}_1$ and $\mathcal{Y}_2$, then $\mathcal{Y}_1$ and $\mathcal{Y}_2$ match with high probability. On the other hand, from the security point of view, in order to be cheat-proof, a good location tag must have the *unpredictability* property. That is, an adversary not at a specific place and time is unable to produce a tag that matches the tag constructed at that location at that time. We can quantitatively measure a location tag's unpredictability through its entropy.

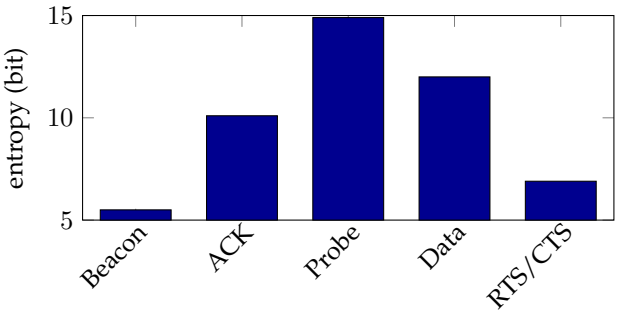
### 4.1 Sources of Location Tags

In our study, we have explored two possible sources of location tags: (1) Using 802.11 MAC headers in WiFi network. MAC frame headers are a family of 34-bit vectors that contain control, duration, address, and sequence control information of the frame. In our assessment, we consider the most common 802.11 MAC headers as shown in Fig. 1**A**. (2) Using radio network temporary identifiers (RNTIs) in LTE networks. RNTIs are a family of 16-bit vectors that are used to differentiate a radio channel or a user from others. There are various types of RNTIs being used in a LTE network. In Fig. 1**B**, we list the ones that can be captured by common mobile devices. Both 802.11 MAC headers and LTE RNTIs are time-varying and location-specific, *i.e.*, users who are observing similar sets of 802.11 MAC headers or LTE RNTIs are likely within the same region at the same time.

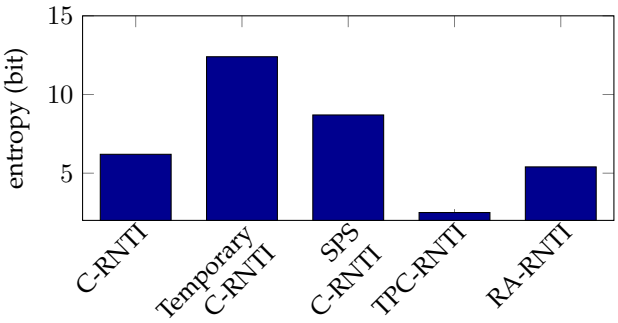
There are two additional reasons for our choices. First, they are the result of a compromise between unpredictability and reproducibility. In our early design, we experimented using full frames as location tag sources. The resulting location tags contain abundant entropy and are highly unpredictable. However, they are difficult to reproduce even

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2015.2472529, IEEE Transactions on Dependable and Secure Computing






(**A**) Entropy of WiFi MAC headers.




(**B**) Entropy of LTE RNTIs.

Fig. 1. (**A**) The length of a 802.11 MAC header is 34 bits. The maximum entropy contained within each header is 14.9 bits. (**B**) The length of a LTE RNTI is 16 bits.The maximum entropy contained within each RNTI is 12.4 bits.

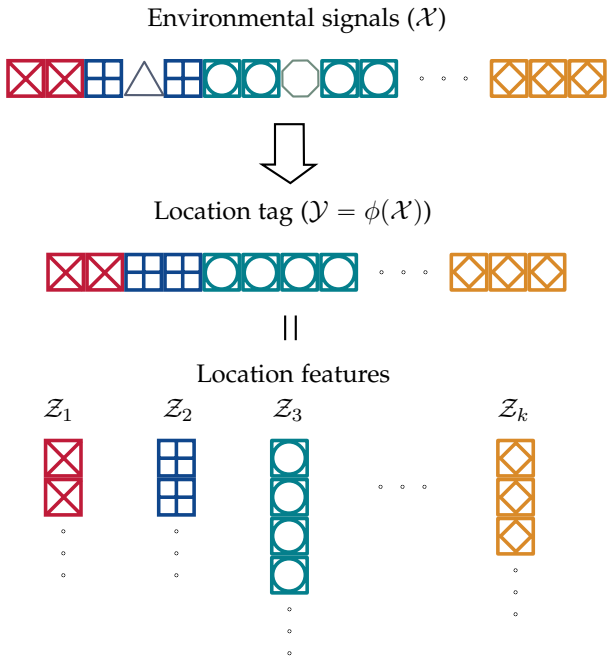



Fig. 2. A location tag is extracted from a set of environmental signals and categorized into different location features for entropy analysis.

for users at the same location. Instead, using only the header or certain fields of a frame allows us to generate reproducible location tags with sufficient entropy. Second, we use the two sources to demonstrate that location tags can be used for proximity test on a different scale. The range of WiFi signal is limited to several hundred metres which allows small vicinity proximity test. The LTE signal can cover an area with a radius of several kilometres which makes them ideal for large vicinity proximity test.

For each location tag, we further divide its observations into multiple groups, $\mathcal{Y} = \{\mathcal{Z}_1, \mathcal{Z}_2, \ldots, \mathcal{Z}_k\}$, where each $\mathcal{Z}$ is defined as a *location feature*. For instance, in Fig. 2, each location feature represents a set of 802.11 or LTE frames of a particular type. The reason for the categorization is to help estimate the amount of entropy contained in the location tag, which we will discuss in detail in Sec. 4.1.

### 4.2 Entropy and Unpredictability

A good location tag should be time-variant and have sufficiently high entropy in order to satisfy the unpredictability requirement. To accurately measure the entropy contained in each location tag, we calculate the entropy rate of the stochastic process [24, p. 269] associated with each location feature. The calculation is done as follows. We consider each location feature, $\mathcal{Z}$, as a stochastic random process defined by a Markov chain. We collect 100,000 observations for each location feature and compute the empirical transition matrix $\mathbf{P}$ of the corresponding Markov chain. Let $Z$ denote the random variables that represent the observations of $\mathcal{Z}$. The transition rate, $p_{ij}$, is defined as

$$p_{ij} = \Pr\left(Z_{k+1} = z_j \mid Z_k = z_i\right).$$

To calculate the entropy, We keep running the Markov chain until it reaches equilibrium. Let $\pi$ denote the stationary distribution of the Markov chain. The entropy of the location feature is

$$H(\mathcal{Y}) = \sum_{ij} \pi_i p_{ij} \log p_{ij}.$$

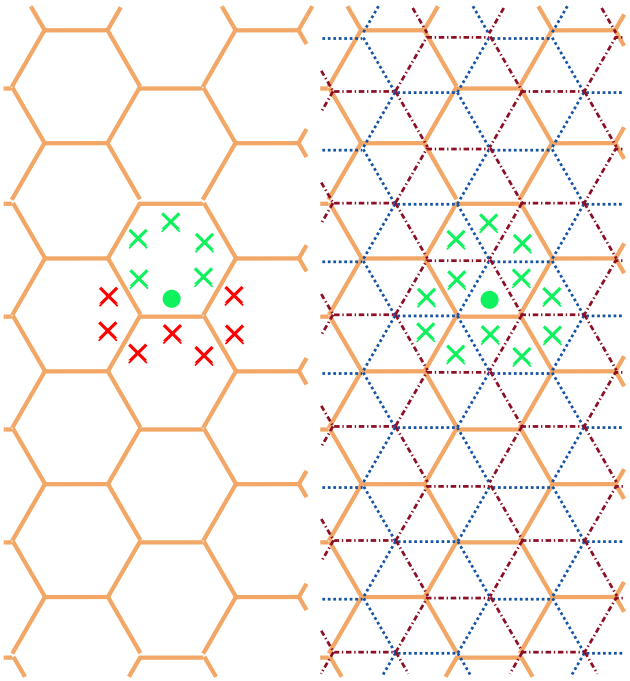
We show the entropy of 802.11 MAC headers in Fig. 1**A**. The headers of beacon frames contain the least amount of entropy since they are transmitted at regular 1024 ms intervals with similar content. The headers of probing request frames on the other hand, contain the most entropy since the algorithm used to scan for access points is not explicitly defined in the 802.11 standard. The interval and format of probing frames are different depending on the device drivers and users' access patterns [25]. In Fig. 1**B**, we show the entropy of LTE RNTIs. Among them, the Temporary C-RNTI contains the highest entropy since the eNodeB issues different Temporary C-RNTIs for the same user during each random access session. Compared to that, the entropy in the C-RNTI is significantly lower due to limited formats or timing variations. The chance that an adversary can predict a location tag with n bits of entropy is $2^{-n}$. Therefore, the entropy contained in both 802.11 MAC headers and LTE RNTIs are sufficient to construct location tags that are unpredictable to adversaries.

## 5 Location Based Handshake and Private Proximity Test

Here we describe the location based handshake and private proximity test protocol. Our protocol is a two-tier procedure designed for a one-to-many proximity test between

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2015.2472529, IEEE Transactions on Dependable and Secure Computing





(**A**) Simplified grid reference. (**B**) Actual grid reference.

Fig. 3. (**A**) On the simplified grid reference, it is possible for points in close proximity, like the green dot and red crosses,to fail the proximity test. (**B**) On the actual grid reference, the former case will pass the proximity test on at least one tessellation.

users who do not have any pre-shared secrets. During the secure handshake Alice initiates the protocol by sending her selection criteria to the server. The server identifies the candidates and coordinates with them to construct their location tags along with Alice. Alice then embeds a temporary key in her location tag and broadcasts it to the candidates. The candidates try to shake hands with Alice by extracting her key using their location tags. During the proximity test, Alice carries out the test with the candidates through a secured communication channel based on the temporary key. To describe our protocol, we first give a brief introduction of the primitives used in the design, *i.e.* Bloom filter, fuzzy extractor, and map grid reference. After clarifying the primitives, we explain the handshake and proximity test protocol steps in chronological order.

### 5.1 Bloom Filter, Fuzzy Extractor, and Map Grid

Here we explain the primitives that are used in our protocol. We utilize the Bloom filter, a space-efficient probabilistic data structure, to succinctly represent location tags. We can formally define a Bloom filter as follows:

***Definition 2.*** Let $y$ be an element in some set. Let $h_i\colon y \rightarrow \{0,1\}^n$ be a hash function. Let $h(y) = \{h_1(y), h_2(y), \ldots, h_k(y)\}$ be a tuple of mutually independent hash functions. A Bloom filter is a $n$-bit vector, $w \in \{0,1\}^n$, equipped with two functions, $\texttt{Ins}\colon (h(y), w) \rightarrow \{0,1\}^n$ and $\texttt{Que}\colon (h(y), w) \rightarrow \{\texttt{FALSE}, \texttt{TRUE}\}$. Let $w_i$ be the $i$th bit of $w$. The insertion function, Ins, sets $w_{h_i(y)}$ to 1, where $1 \leqslant i \leqslant k$. The query function, Que, returns TRUE if $w_{h_i(y)} = 1$, where $1 \leqslant i \leqslant k$, and FALSE otherwise.

When using Que for a membership test, false positives are possible due to the collision between hash functions, but false negatives are not. In our case, we represent a location tag by inserting all the observations into an empty Bloom filter.

We construct a Bose-Chaudhuri-Hocquenghem (BCH) code based fuzzy extractor [10] for the key embedding/extraction operations. It allows users, whose location tags are similar but not identical to Alice's location tag, to extract Alice's temporary session key. Generally, a fuzzy extractor can be defined as follows:

***Definition 3.*** Let $\mathcal{W}$ be the input space equipped with a metric function $\mathrm{dis}\colon (w \in \mathcal{W}, w' \in \mathcal{W}) \rightarrow \mathbb{R}$. A $(\mathcal{W}, m, \ell, t, \epsilon)$-fuzzy extractor is a pair of randomized functions, $\texttt{Gen}\colon w \in \mathcal{W} \rightarrow \left(r \in \{0,1\}^\ell, s \in \{0,1\}^*\right)$ and $\texttt{Rep}\colon (w' \in \mathcal{W}, s) \rightarrow r' \in \{0,1\}^\ell$, with the following two properties. (1) The reproduction, $r'$, equals the original, $r$, if $\mathrm{dis}(w, w') \leqslant t$. (2) For any distribution on $\mathcal{W}$ of min-entropy $m$ [10], the distribution of $r$ is $\epsilon$-close [10] to uniform even to those who observe $s$.

Intuitively, a fuzzy extractor allows one to extract some randomness $r$ from $w$ and then successfully reproduces $r$ from any $w'$ that is close to $w$. Most fuzzy extractors can be constructed based on error-correcting codes. In our case, we use the BCH code based construction, which can be decoded in polynomial time. In our scheme, the value $r$ represents Alice's temporary session key; the value $s$ represents the key embedding; the inputs $w$ and $w'$ represent the Bloom filter vectors that store the location tags.

Finally, in order to carry out the fine-grained proximity test, the system adopts a *grid reference* beside location tags to represent users' locations. As shown in Fig. 3**A**, let $g$ be a grid or a hexagon tessellation of the Earth's surface. A user's position is represented by a grid block $g_i$. Note that we simplify the grid tessellation to ease the visual representation. In practice, it is common to use multiple tessellations which are mutually offset, as shown in Fig. 3**B**, to circumvent the cases when two users are close but on different sides of a partition as shown in Fig. 3**A**.

### 5.2 Location Based Handshake

Fig. 4 describes the first phase of our handshake and proximity test protocol. It starts by Alice sending a request message

$$\texttt{REQ} = \{\varphi(\cdot), \tau_1, \tau_2, \phi(\cdot), h(\cdot)\}$$

to the server. The request contains a filtering function, $\varphi(\cdot)$, that specifies Alice's selection criteria of the candidates, two timestamps $\tau_1$ and $\tau_2$ that specify the starting and ending times on which to synchronize the generation of the location tags, a filtering function, $\phi(\cdot)$, that specifies the filtering rules of the location tag, and a hash function tuple, $h(\cdot)$.

Upon receiving REQ, the server identifies the candidates based on $\varphi(\cdot)$ and broadcasts a synchronization message

$$\texttt{SYN} = \{\tau_1, \tau_2, \phi(\cdot), h(\cdot)\}$$

to all candidates.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2015.2472529, IEEE Transactions on Dependable and Secure Computing




Fig. 4. Alice embeds her temporary public key into her location tag and broadcasts the embedding messages to all candidates. Only nearby users with locations tags similar to Alice's location tag can extract her public key.

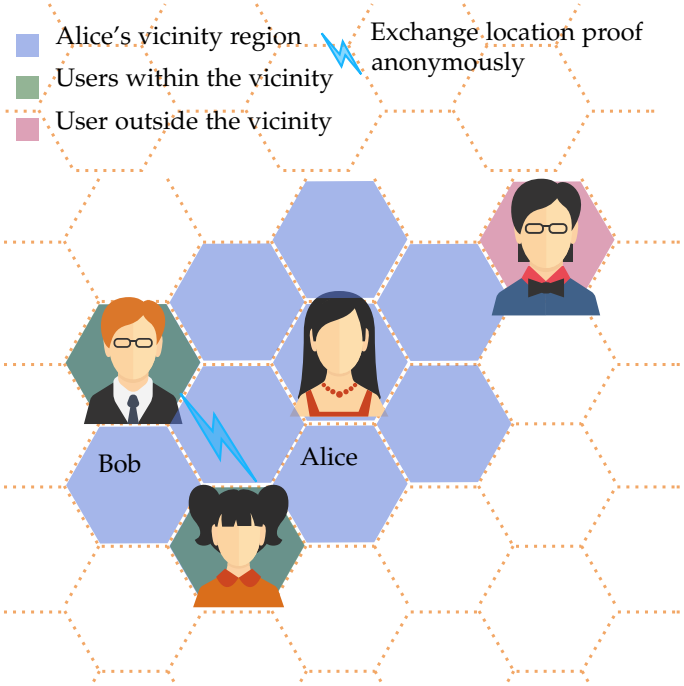



Fig. 5. Alice performs the private proximity test by allowing users to query the Bloom filter that represents her vicinity region using her temporary public key. Additionally, Alice requires users to submit the $eid$s they collected as location proof.

After receiving SYN, each user collects a set of environmental signals $\mathcal{X}(\tau_1, \tau_2)$ falling between the specified timestamps, and extracts a location tag $\mathcal{Y}(\tau_1, \tau_2) = \phi(\mathcal{X}(\tau_1, \tau_2))$ by applying the filtering function $\phi(\cdot)$. To store the location tag, each user initializes an empty Bloom filter, $w = \{0\}^n$, and inserts the observations, $y_i(\tau_1, \tau_2)$, into $w$ through the insertion function of the Bloom filter

$$w = \texttt{Ins}(h(y_i(\tau_1, \tau_2)), w).$$

As shown in Fig. 6**A**, we use the code-offset construction for fuzzy extractors [10] by Dodis *et al.* to embed a key into Alice's location tag. Let $w_A \in \{0,1\}^n$ be the Bloom filter that represents Alice's location tag. Alice computes a pair of RSA keys, $r_{SK}$ and $r_{PK}$. Alice encodes the public key, $r_{PK} \in \{0,1\}^\ell$, using a BCH encoder over a Galois field,

$$c = \texttt{Encode}(n, \ell, r_{PK}) \in GF(2^n),$$

and the shift needed to get from $c$ to $w_\text{A}$

$$s = c - w_A \in \{0,1\}^n.$$

Alice then sends the embedding message

$$\texttt{EMB} = \{s\}$$

to the server.

The server broadcasts EMB to the candidates. Each candidate can try to extract $r_{PK}$ using their location tag. Let $w_B$ be the Bloom filter that represents the location tag of a candidate, Bob. To extract $r_{PK}$, Bob computes

$$c' = s - w_B$$

and tries to decode the message using a BCH decoder,

$$r'_{PK} = \texttt{Decode}(n, k, c').$$

Based on the property of BCH codes, $r_{PK} = r'_{PK}$ only if $\texttt{Ham}(c, c') \leqslant t$, where $t$ is the maximum number of bits that BCH coding can correct. Since $w_A = c - s$ and $w_B = c' - s$, it means the difference between $w_A$ and $w_B$ is at most $t$ bits. Otherwise, Bob cannot retrieve $r_{PK}$. Note that Bob cannot verify whether he has retrieved $r_{PK}$. The reason is that, when the number of errors exceeds $t$, the BCH decoder may unknowingly produce an apparently valid message that is not the one that was sent. Therefore, Bob cannot know whether he is within the coverage of Alice's location tag.

### 5.3 Private Proximity Test

Fig. 5 describes the second phase of our handshake and proximity test protocol. Once the handshake is completed, Alice can perform a private proximity test with the candidates. Alice first defines her vicinity region on the grid map, $g$. Note that the shape of the vicinity region can be arbitrary. For instance, Alice can define her vicinity region to be a disk with radius $\delta$, or the building that she is currently in. Let $\mathbf{g}(A)$ be the minimal set of grid blocks that covers Alice's vicinity region. As shown in Fig. 6**B**, Alice initializes another empty Bloom filter with a vector, $\hat{w} = \{0\}^n$, and inserts each grid block index into $\hat{w}$ by applying

$$\hat{w} = \texttt{Ins}(h(g_i(A) \,\|\, r_{PK}), \hat{w}),$$

where $g_i(A) \,\|\, r_{PK}$ is the concatenation of a grid block index and Alice's secret key. Alice then sends a proximity test message $\hat{w}$ to the server.

$$\texttt{PRO} = \{\hat{w}\},$$

To answer Alice's proximity test. Bob first needs to exchange location proofs with other users. Let $id_B$ be Bob's

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2015.2472529, IEEE Transactions on Dependable and Secure Computing



ID which consists of his device's MAC address signed by his private key. To answer the query, Bob encrypts $id_B$ with Alice's public key using RSA public key encryption

$$eid_B = \texttt{Encrypt}(id_B, r'_{PK}),$$

and broadcast $eid_B$ through WiFi or Bluetooth. The range of the broadcast signal must be smaller than the coverages of location tags. For WiFi signal, Bob may rely on the nearby access points to help broadcast his $eid$. Meanwhile, Bob collects the $eid$s shared by other users. Let Bob's location proof, $\mathcal{P}_B$, be a set of $eid$s collected by him. Bob then identifies a grid block, $g(B)$, that represents his current location and sends a response message

$$\texttt{RSP} = \{h(g(B)\|r'_{PK}), eid_B, \mathcal{P}_B\}$$

to the server.

The server can perform the first round filtering for Alice using the Bloom filter query function

$$\texttt{Que}(h(g(B)\|r'_{PK}), \hat{w}).$$

There are two possibilities when the query returns FALSE, either Bob cannot extract $r_{PK}$ correctly or his current location is beyond Alice's vicinity region. If the query returns TRUE, Bob must be within the coverage of Alice's location tag. However, Bob can be dishonest about $g(B)$ since he can pick any grid block that is within the coverage of Alice's location tag. To rule out potential liars, the server constructs a directional graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$, in which, each vertex, $v$, is the $eid$ of a user who passes the first round filtering; each arc, $a$, points from the user to one of the $eid$s in his location proof. Let $\mathbf{G}$ be the adjacency matrix of $\mathcal{G}$. The server divides $\mathcal{G}$ into subgraphs by applying the eigenvector-based partition method [26] to $\mathbf{G}$ with a maximum number of $|\mathbf{g}(A)|$ subgraphs. For each subgraph, the server eliminates the vertices that have significantly fewer incoming arcs than outgoing arcs. Those vertices represent the users who are not seen by the peers they claim to see in their location proofs. Let RST be the set of remaining $eid$s. The server sends RST to Alice. Alice decrypts each $eid$ in RST by applying

$$id = \texttt{Decrypt}(eid, r_{SK})$$

and obtains a set of identities who are within her vicinity region. To secure subsequent communication with these these users, Alice can retrive their public keys from a certificate authority (CA) base on their identities.

## 6 Analysis

Here we analyze the functionality, security, and privacy properties of our handshake and proximity test protocol. We show that the protocol is applicable to different settings based on the relation between the coverage of the location tags and the size of the grid blocks. To demonstrate the protocol is robust against location cheating, we consider two scenarios: (1) Bob is beyond the coverage of Alice's location tag. (2) Bob is within the coverage of Alice's location tag but is outside of Alice's vicinity region. In both cases, we show that Bob cannot fabricate his location to trick Alice. We assess the privacy property of the protocol. We show the three privacy guarantees of our protocol: (1) the server is oblivious of Alice's location and Bob's location throughout the protocol. (2) Alice only learns whether Bob is within her vicinity region and nothing else. (3) Bob does not know Alice's location even if he is within Alice's vicinity region. Finally, we consider the efficiency of the protocol and the possibility of denial-of-service (DoS) attack.

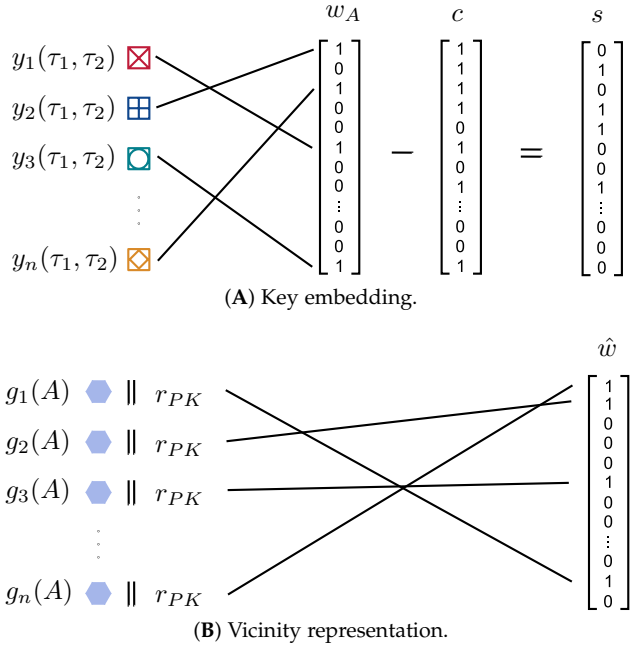



Fig. 6. (**A**) The key embedding is created by inserting a location tag's observations into a Bloom filter and taking the exclusive or of the Bloom filter vector and the BCH codeword of the key. (**B**) Alice's vicinity region is represented by inserting each concatenation of a grid block and the key into a Bloom filter.

### 6.1 Functionality

One of the drawbacks of location tags is that the users cannot control the precise range of the environmental signals. For instance, if Alice chooses to use WiFi signals to construct her location tag, the signal range is no larger than a few hundred metres. If Alice chooses to use LTE signals, her location tag could cover the area with a radius of several kilometres. If Alice constructs her location tag using a combination of environmental signals, its coverage is usually determined by the shortest range signals. Since the coverage of the location tag correlates to the number of users who can handshake with Alice, location tags of different coverages could affect the functionality of the protocol.

Let $l_{\mathcal{T}}$ be the range of the environmental signals used to construct the location tags. Let $l_g$ be the size of a grid block. Let $l_v \geqslant l_g$ be the size of Alice's vicinity region. Ideally, Alice should use a location tag such that $l_{\mathcal{T}} \simeq l_v$. This way the location based handshake can effectively filter out a larger number of users who are far away from Alice, which reduces the number of potential liars in the second phase. The proximity test then, is only used for fine-grained tuning based on the shape of Alice's vicinity region. For instance, Alice can use McDonald's' WiFi signals for the handshake, and uses the proximity test to identify users who are in fact within the MacDonald's rather than the adjacent Wendy's.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2015.2472529, IEEE Transactions on Dependable and Secure Computing



In practice, however, it is usually the case that $l_{\mathcal{T}} > l_v$. For instance, Alice considers the users who are under the same LTE cell tower to have the same location tag as her whereas her vicinity region is only the street block she is currently located. Our protocol is applicable in such a scenario. The only difference is that the number of users who can get Alice's public key increases, which increases the chance of a collusion attack. We will address this problem in Sec. 6.2.

Finally, there is the improbable case that $l_v > l_{\mathcal{T}} > l_g$. A example of such a case would be Alice using femtocell signals as her location tag and considering the entire campus as her vicinity region. Our protocol will return incomplete results that consist of only users that are within the same femtocell as Alice. However, since Alice is in control of the size of the vicinity region and the choice of the environmental signals, it is unlikely that Alice would choose such an unreasonable setting.

### 6.2 Security

Here we analyze the security properties of our protocol. In particular, we show that our protocol is robust against location cheating. Assume Bob wishes to lie about his location. We consider Bob achieves location cheating if one of the following two scenarios holds: (1) Bob has a location tag that is distinct from Alice's location tag. but Bob successfully extracts Alice's public key, $r_{PK}$. (2) Bob's grid block, $g(B)$ is outside of Alice's vicinity region, $\mathbf{g}(A)$, but Bob convinces Alice that $g(B)$ is within $\mathbf{g}(A)$.

Let $\mathcal{T}_A$ be Alice's location tag. Let $\mathcal{T}_B$ be Bob's location tag. The first scenario implies that

$$|\mathcal{T}_A \triangle \mathcal{T}_B| > t_{\mathcal{T}},$$

where the symbol $\triangle$ represents the symmetric difference between two sets and $t_{\mathcal{T}}$ is the difference threshold set by Alice. However, due to the false positives of Bloom filters, there is a possibility that the Hamming distance of the corresponding Bloom filters is less than $t = kt_{\mathcal{T}}$, *i.e.*,

$$\texttt{Ham}(w_A, w_B) \leqslant t. \tag{1}$$

In that case, Bob can extract $r_{PK}$ even though he is far away from Alice. However, the probability of false positive can be controlled by Alice. Assuming $|\mathcal{T}_A| \simeq |\mathcal{T}_B|$, the probability of a certain bit being 1 after inserting $|\mathcal{T}_A|$ elements is

$$1 - \left(1 - \frac{1}{n}\right)^{k|\mathcal{T}_A|}.$$

Therefore, the probability that Eq. 1 holds is approximately

$$\sum_{i=k|\mathcal{T}_A \triangle \mathcal{T}_B|-t}^{k|\mathcal{T}_A \triangle \mathcal{T}_B|} \left(1 - \left(1 - \frac{1}{n}\right)^{k|\mathcal{T}_A|}\right)^i.$$

By increasing the size of the Bloom filter, Alice can reduce the probability that Bob accidentally extracts her public key due to the collision of the the hash values. Assuming that the attacker's Bloom filter is constructed based on genuine environmental signals, the chance to find a location such that the location tag's hash values collide with the hash values of Alice's location tag is slim. On the other hand, if the attacker chooses to disregard the protocol and fabricates his Bloom filter directly, he will need to perform a dictionary attack and try all $2^n$ possibilities.

Additionally, the symmetric difference based matching strategy is robust against the attack where Bob uses a high-gain antenna to extend the coverage of his location tag. Assuming all honest users have devices with typical antenna gain. Bob can employ a high-gain antenna to sense environmental signals at a remote distance and extend the coverage of his location tag. However, since Bob's location tag now contains all environmental signals within a larger area, the symmetric difference between Bob's location tag and Alice's location tag does not necessarily decrease [27]. Therefore, Bob still cannot extract Alice's key. The same argument holds if multiple adversaries try to extract Alice's key by combining their location tags.

In the second scenario, Bob is already within the coverage of Alice's location tag. He can obtain Alice's public key legitimately using his true location tag. However, Bob is outside of Alice's vicinity region, *i.e.*,

$$g(B) \notin \mathbf{g}(A).$$

To trick Alice, Bob uses another grid block $g'(B) \in \mathbf{g}(A)$ and responds with $h(g'(B) \| r'_{PK})$. The Bloom filter query, $\texttt{Que}(h(g(B) \| r'_{PK}), \hat{w})$ will return `TRUE`. However, since Bob fails to provide a location proof, $\mathcal{P}_B$, that identifies other users present in $g'(B)$, the server tends to mark him for location cheating. Bob may attempt a Sybil attack by creating a large number of pseudonymous identities to provide mutual proofs. However, for LBS, such attack can be prevented by directly associating users' identities with their social network accounts [28], which raises the barrier to create Sybil identities. In that case, the social network provider operate as a CA, and verifies Bob's identity information, such as social security number (SSN), before issuing his private key. This way, Alice can verify the identities of the users who pass the proximity test by pulling their public keys from the CA. Since social networking applications are commonly integrated into users' smartphones, our Sybil defense method can work seamlessly with our location proximity scheme. Note that it is still possible for multiple adversaries who are within the coverage of Alice's location tag to collude, and use mutual proofs to trick the server. In that case, the server will see multiple anonymous subgraphs for the same grid block. To identify the colluding group, the server can use certified location information for the proximity test [29], which could potentially violate users' privacy. Finally, to further prevent impersonation and eavesdropping after the proximity test, we require Alice to utilize users' public keys on top of $r_{PK}$ to secure each communication channel. This way, users who pass the proximity test cannot overhear each other when communicating with Alice.

### 6.3 Privacy

Our protocol provides strong privacy assurances for both Alice and Bob. For Alice, her location is represented by a fuzzy extractor helper string, $s$, and a Bloom filter, $\hat{w}$. Since the server is not physically present anywhere near Alice, it cannot obtain Alice's public key, $r_{PK}$, and use it to query $\hat{w}$. Therefore, the server remains oblivious of Alice's location throughout the protocol. Bob, on the other hand, cannot tell

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2015.2472529, IEEE Transactions on Dependable and Secure Computing



whether he has extracted $r_{PK}$ correctly, due to the nature of the BCH decoder. Therefore, he cannot locate Alice based on whether he is within the coverage of Alice's location tag.

For Bob, his location is represented by $k$ hash values, $h(g'(B)\|r'_{PK})$. Based on the same argument, the server cannot learn Bob's location without Alice's public key, $r_{PK}$. Additionally, since the server does not forward $h(g'(B)\|r'_{PK})$ to Alice, if the proximity test returns positive, Alice only learns Bob's identity without knowing his exact location.

However, our protocol can still leak users' privacy if users collude with the server. If Alice colludes with the server, Alice might learn Bob's exact location if Bob is within the coverage of her location tag. If Bob colludes with the server, Bob might learn Alice's vicinity region if he is within the coverage of Alice's location tag. But, when Alice and Bob are far apart, *i.e.*, the difference between their location tags is significant, the user-server collusion cannot breach users' location privacy.

### 6.4 Efficiency

Finally, we analyze the protocol efficiency. Our protocol is designed to minimize the effort at the users' side. Let $N$ denote the number of candidates in the proximity test. For Alice, the main computation is evaluating the hash values of her location tag and encoding the BCH codeword. Therefore, the computation complexity for Alice is $\mathcal{O}(k\,|\mathcal{T}_A| + n)$. The size of the location tag, $|\mathcal{T}_A|$, depend on the signal traffic density at Alice's location, whereas the length of the BCH codeword, $n$, is controlled by Alice. In practice, we can assume that $|\mathcal{T}_A|$ and $n$ are constant, which gives a $\mathcal{O}(1)$ computational complexity in respect to $N$ at Alice's side. For Bob, the main computation is evaluating the hash values of his location tag and decoding the BCH codeword. If we use the classic Berlekamp-Massey decoder [30], the computational complexity for Bob is bounded by $\mathcal{O}(k\,|\mathcal{T}_B| + n^2)$. If we again assume that $|\mathcal{T}_B|$ and $n$ are constant, the computational complexity at Bob's side is also $\mathcal{O}(1)$ in respect to $N$.

The computational cost at the server's side is dominated by the number of candidates, $N$, assuming all candidates are willing to participate in the test. Since Bob cannot tell whether he can extract Alice's public key, he always sends a `RSP` message to the server even if he does not have the correct key. The server, therefore, always has $N$ `RSP` messages to process. For each `RSP` message, the server needs to perform a Bloom filter membership query which costs $\mathcal{O}(1)$ in respect to $N$. Finally, let $N' \ll N$ be the number of remaining candidates who pass the first round filtering. The server needs to perform an eigenvalue decomposition of the adjacency matrix $\mathbf{G} \in \{0,1\}^{N'\times N'}$, which costs $\mathcal{O}(N'^3)$ if $\mathbf{G}$ is dense, or $\mathcal{O}(N')$ if $\mathbf{G}$ is sparse.

From the analysis, we see that Alice may launch a DoS attack by querying a large number of candidates using a broader filtering criteria. To mitigate such problem, we can employ a common technique in database management, and let Alice select $\varphi(\cdot)$ from a set of predefined filter functions to limit the size of $N$. Additionally, the location based handshake further reduces the chance of DoS attack from Alice. Otherwise, the server needs to compute an eigenvalue decomposition of an adjacency matrix of size $N \times N$, which could cost up to $\mathcal{O}(N^3)$.

## 7 Evaluation

We have carried out both simulation and real-word experiments to assess the feasibility and performance of our protocol. To assess its feasibility, we implemented the handshake and proximity test protocol and created two hypothetical scenarios simulating WiFi network and LTE network planning. We evaluated the spatial-temporal properties of location tags and the possibility of location cheating using the two scenarios. To test its performance, we used wireless protocol sniffers to collect real-world WiFi and LTE traces, based on which we estimated the size and construction time of location tags as well as the impacts of synchronization and mobility to the handshake.

### 7.1 Tools

Here we briefly introduce the software and hardware we used for our evaluation. We used `Wireless InSite` [31] to estimate the radio coverage of WiFi and LTE networks. `Wireless InSite` is a a suite of ray-tracing models and EM solvers for the analysis of site-specific radio propagation. It allows us to simulate the coverage of a given wireless network. We used `ns-3` [32] to simulate the packet delivery of WiFi and LTE networks. `ns-3` is a a discrete-event network simulation platform. It contains various models which allow us to examine the connectivity of a given wireless network. We used `Riverbed AirPcap` [33] and `IntelliJudge` [34] to capture real-world WiFi and LTE traces. `Riverbed AirPcap` is a WiFi packets sniffer which allows us to record packets from WiFi network. `IntelliJudge` is a LTE sniffing tools which allows us to record and decode messages from all LTE layers.

### 7.2 Simulation Setup

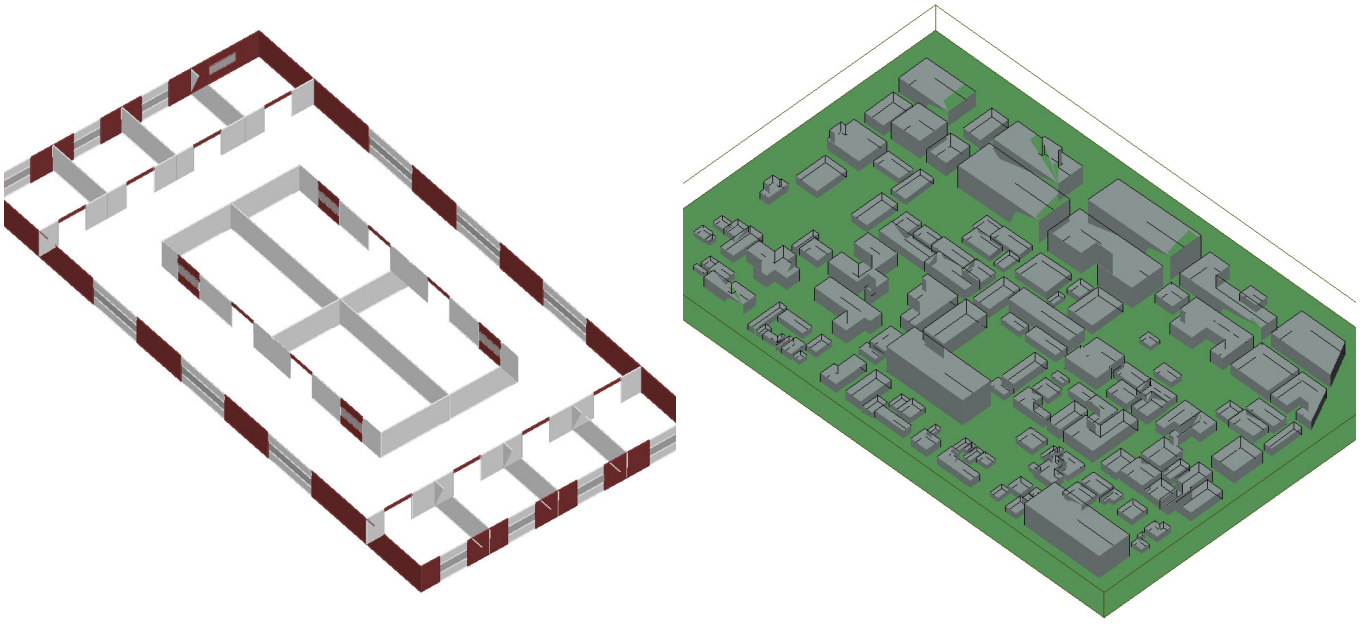
As shown in Fig. 7, we created two wireless network planning scenarios. The WiFi network is based on the access points deployed in a typical office building; the LTE network is based on the LTE macrocells deployed in an urban area. For the WiFi scenario, we considered an area of $5000\,\mathrm{m}^2$ with a vicinity region of $400\,\mathrm{m}^2$ (the radius is approximately $14\,\mathrm{m}$) and a grid block of $100\,\mathrm{m}^2$. For the LTE scenario, we considered an area of $12.5\,\mathrm{km}^2$ with a vicinity region of $1\,\mathrm{km}^2$ (the radius is approximately $700\,\mathrm{m}$) and a grid block of $40{,}000\,\mathrm{m}^2$. We set the free space signal range of LTE, WiFi, and Bluetooth to $1\,\mathrm{km}$, $100\,\mathrm{m}$, and $5\,\mathrm{m}$ respectively. We tested each scenario with different population densities. For each setting, we ran 100 Monte Carlo simulations and took the average.

### 7.3 Simulation Results

Here we show the simulation results. We used the BCH(511,202) code to construct our fuzzy extractor, which can tolerate up to 42 symmetric differences between two 202 bit vectors. Since the size of the RSA public key is larger than the message length, $\ell$, we padded the key with 0s and divided it into multiple blocks of 202 bits before encoding. Fig. 8 shows the symmetric similarities of simulated location tags. The gray lines mark the thresholds where the BCH code cannot help recover Alice's public key. The range of LTE location tag agrees with the free space propagation

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2015.2472529, IEEE Transactions on Dependable and Secure Computing



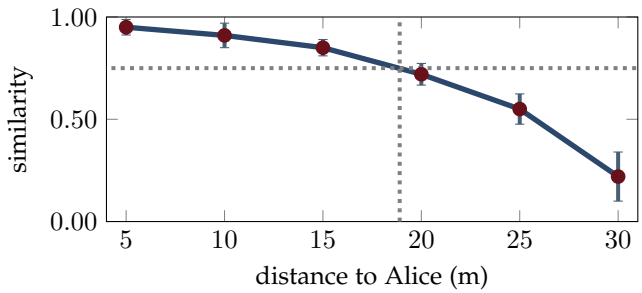


(**A**) WiFi scenario. (**B**) LTE scenario.

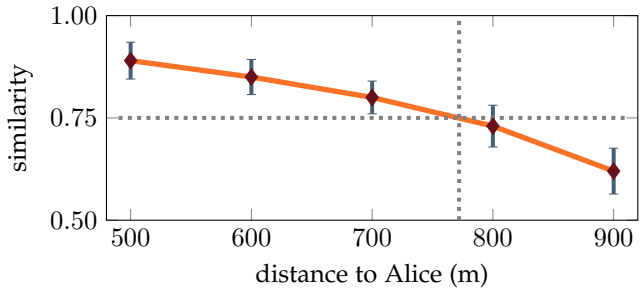
Fig. 7. (**A**) The floor plan we used for the WiFi scenario. The study area is $50\,\mathrm{m}$ by $100\,\mathrm{m}$. For each Monte Carlo simulation, we randomly placed 10 access points within the study area. (**B**) The building data we used for the LTE scenario. It is based on the example provided by `Wireless Insite` [31]. The study area is $5\,\mathrm{km}$ by $2.5\,\mathrm{km}$. For each Monte Carlo simulation, we randomly placed 3 LTE cell towers within the study area.

model of LTE signals since the LTE scenario is based on an outdoor environment with line-of-sight (LOS) transmission paths. The range of the WiFi location tag, on the other hand, is significantly shorter than the corresponding free space model due to the obstructions of an indoor environment. For both WiFi and LTE signals, the similarities decrease as the candidates become further away from Alice, which shows that it is viable to achieve location based handshaking with ambient radio signals.

We evaluated the possibility of location cheating through a simulated adversary who tries to convince Alice that they are within the same grid block. We varied the population densities of the simulated scenarios to verify the robustness of the ad hoc location proof exchanges against location cheating. We used WiFi and Bluetooth signals to exchange location proofs in the LTE and WiFi scenario respectively. As shown in Fig. 9, the adversary cannot conduct location cheating when he is beyond the coverage of Alice's location tag. When the adversary passes the location based handshake, the degree of security against location cheating varies by the population density. When there are multiple people within the range of the $eid$ broadcasting signals, the server can identify the liar with high confidence. However, when the population density decreases, the maximum size of the clique on the $eid$ graph decreases. As a result, it lowers the server's capability to discern location cheaters within the coverage of Alice's location tag. However, the server can report truthfully that it cannot determine the authenticity of claimed locations when the connectivity of the $eid$ graph is less than a predefined threshold. In that case, Alice may require users to submit certified location information [29] for the proximity test.




(**A**) WiFi location tag.




(**B**) LTE location tag.

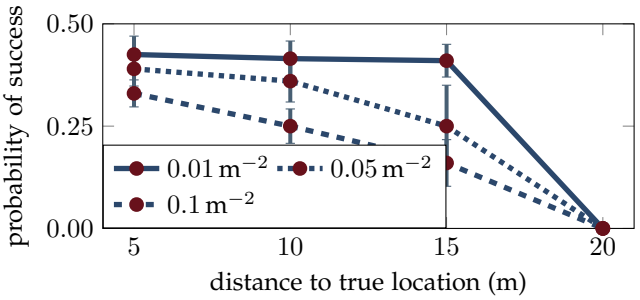
Fig. 8. (**A**) The WiFi location tags can help candidates within an $18\,\mathrm{m}$ radius to recover Alice's public key. (**B**) The LTE location tags can help candidates within a $781\,\mathrm{m}$ radius to recover Alice's public key.

### 7.4 Experiment Setup

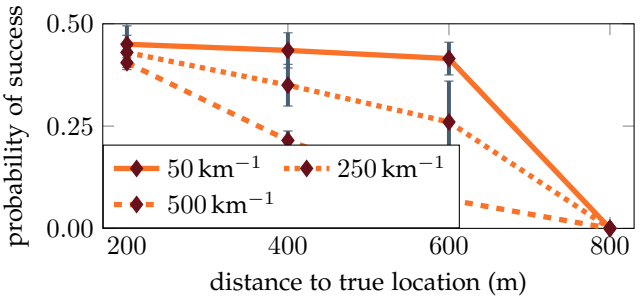
We performed two experiments to evaluate the location based handshake using location tags in the real-world. The experiment on WiFi location tags was conducted within

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2015.2472529, IEEE Transactions on Dependable and Secure Computing



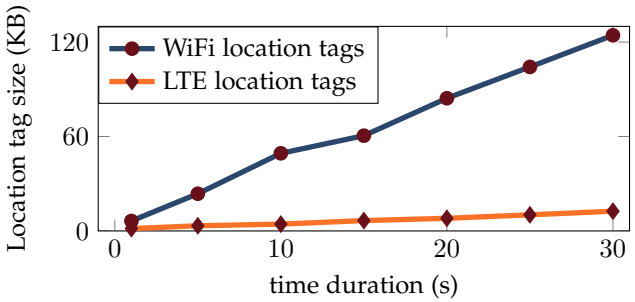



(**A**) WiFi location tag.

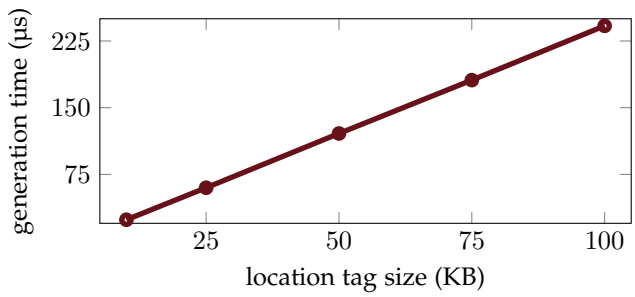



(**B**) LTE location tag.

Fig. 9. (**A**) The possibility of location cheating versus population densities in the WiFi scenario. (**B**) The possibility of location cheating versus population densities in the LTE scenario.




(**A**) Location tag size.




(**B**) Location tag generation time.

Fig. 10. (**A**) The size of location tags grows linearly to the capture time to reduce the case of false positive during the handshake. (**B**) The generation time of location tags grows linearly to the size of location tags

our department building. The experiment on LTE location tags was conducted within our university's campus area. We set up three laptops to capture the WiFi or LTE traffic. We estimated the ranges of WiFi and LTE location tags to be 20 m and 1 km respectively. For each capture, we randomly placed one laptop within the estimated ranges (approximately 15 m for WiFi location tag; approximately 500 m for LTE location tag) and one beyond the estimated ranges (approximately 30 m for WiFi location tag; approximately 1.5 km for LTE location tag). We used synchronized clocks to coordinate traffic capture. We considered it a false positive when the distant laptop could extract Alice's key. We considered it a false negative when the nearby laptop failed to extract Alice's key. We ran each experiments 30 times to calculate the false positive rate and false negative rate.

If we fix the traffic capture durations, the size of the Bloom filter vector can vary in order to minimize the number of false positive cases during the handshake. Fig. 10**A** shows the optimal size of the Bloom filter that yields a false positive rate[2] of $FPR = 1^{-10}$. The size of the Bloom filter, $|w|$, is computed as [35]

$$|w| = \lceil |\mathcal{Y}| \frac{\log 1/FPR}{\log^2 2} \rceil.$$

Additionally, we evaluated the CPU time required to generate a location tag. Fig. 10**B** shows location tag generation time on a Google Nexus 7 with 1.5 GHz quad-core Snapdragon S4 Pro processor. The hash functions we used are based on SHA-256 hash functions that benchmark 3.5 ns on the device. The number of hash functions, $k$, is computed as [35]

$$k = \lceil |w| \frac{\log 2}{|\mathcal{Y}|} \rceil.$$

The majority of CPU time is used to insert observations into the Bloom filter and encode the Bloom filter through the BCH encoder. The compuational complexity to generate a location tag grows linearly to the number of observations.

In practice, it is easier to fix the size of the Bloom filter and vary the capture durations to minimize the number of false positive cases during the handshake. We identified the optimal duration that yields the minimal false positive rate for the BCH(511,202) based location tags. As shown in Fig. 11, the optimal durations to capture WiFi traffic and LTE traffic are about 10 s and 20 s respectively. If the capture durations are shorter, the Bloom filter becomes emptier, which reduces the symmetric difference and increases the false positive rate of the handshake. If the capture durations are longer, the Bloom filter becomes fuller, which also reduces the symmetric difference and increases the false positive rate of the handshake. Within the optimal durations, we captured an average of 8432 WiFi MAC headers and 1375 LTE RNTIs. Half of the WiFi MAC headers belong to Beacon packets. The rest of the packets are ACKs, Probe responses, Probe requests, etc. The majority of LTE RNTIs are from the C-RNTI family.

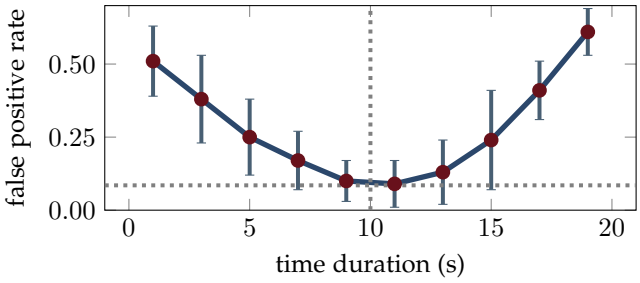
### 7.5 Experiment Results

Here we show our experimental results from the real-world location tags. The metrics we use to evaluate the protocol's performance are the areas under the curve (AUC) of the

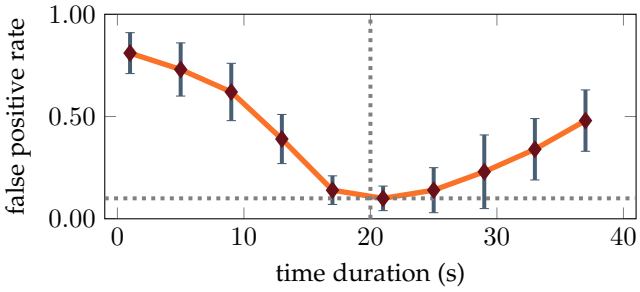
2. Note that the false positive rate of the Bloom filter is associated with but not equal to the false positive rate of the handshake.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2015.2472529, IEEE Transactions on Dependable and Secure Computing






(**A**) WiFi location tag.




(**B**) LTE location tag.

Fig. 11. (**A**) The WiFi MAC headers can rapidly fill up the Bloom filter, which leads to a high false positive rate during the handshake. (**B**) The LTE RNTIs contains less entropy and fill up the Bloom filter more gradually, which increases the optimal capture duration.

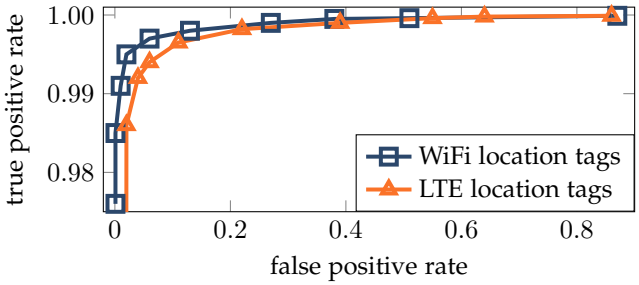
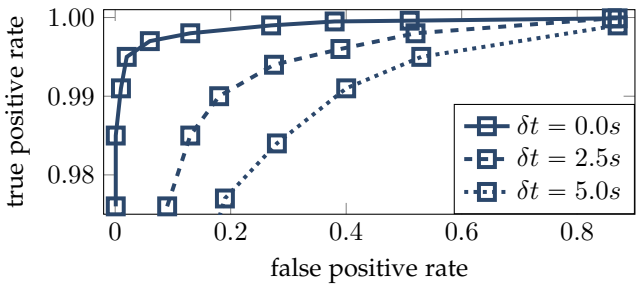



Fig. 12. The high ROC AUCs of both WiFi and LTE location tags suggest that location tags are ideal for location based handshakes.

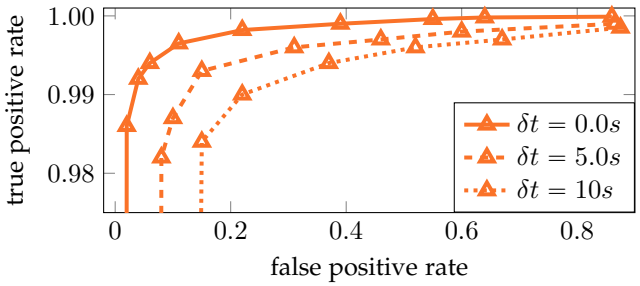
receiver operating characteristic (ROC) curves. We generate the ROC curves by varying the parameters of the BCH code. The most tolerable BCH code we used is the BCH(512, 10) code which can correct up to 121 errors within a 10 bit message. The least tolerable BCH code we used is the BCH(512, 502) code which can correct up to 1 error within a 502 bit message.

Fig. 12 shows the ROC curves of WiFi and LTE location tags. The ROC AUC of WiFi location tags is approximately 0.993 whereas ROC AUC of LTE location tags is approximately 0.992. It shows that location based handshake with location tags is an effective method to filter out candidates who are far away from Alice. The ROC AUC of LTE location tags are slightly lower than WiFi since LTE RNTIs contains less entropy than WiFi MAC headers. Therefore, LTE location tags are more prone to hash collisions.

**Clock Synchronization Error and Reproducibility**. We tested the protocol's performance against clock synchronization errors. We purposely configured a time delay, $\delta t$, between Alice and the other two laptops during each




(**A**) WiFi location tag.




(**B**) LTE location tag.

Fig. 13. The ROC AUCs of both WiFi and LTE location tags decrease when synchronization errors occur.

capture. As shown in Fig. 13, when users do not start the traffic capture process simultaneously, the average difference between location tags increases, which decreases the ROC AUCs. However, the decrease is minimal compared to the ratio between the time delay and capture duration. Under our experimental setup, when $\delta t$ is half the length of the capture duration, the ROC AUCs decrease by 10.5 % and 7.6 % for the WiFi and LTE location tags respectively. Compared to LTE location tags, the ROC AUC of WiFi location tags decrease more drastically. It is partly due to the shorter duration we used to capture WiFi traffic and the fact that WiFi traffic contains more entropy which is harder to reproduce at a different time. Therefore, by increasing the BCH codeword length, we can increase the optimal capture duration and protocol's tolerance to synchronization error.

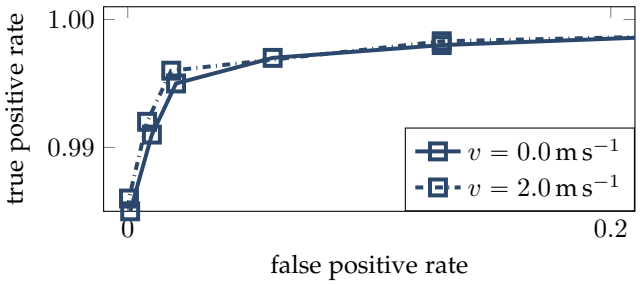
**Mobility and Reproducibility**. We evaluated how mobility affects the performance of the protocol. In the experiment, we allowed each candidate laptop to move at an average pedestrian velocity. Compared with the stationary case, the ROC AUCs for both WiFi and LTE location tags witness a slight bump. The reason is that a moving candidate is able to capture more traffic than a stationary candidate, which increases the chance of handshake. The mobility effect is more drastic on WiFi location tags than LTE location tags. It is because the range of WiFi signal is much smaller than the range of LTE signal. However, the advantage a candidate gains by moving is minimal since each capture window is short.

## 8 Discussion

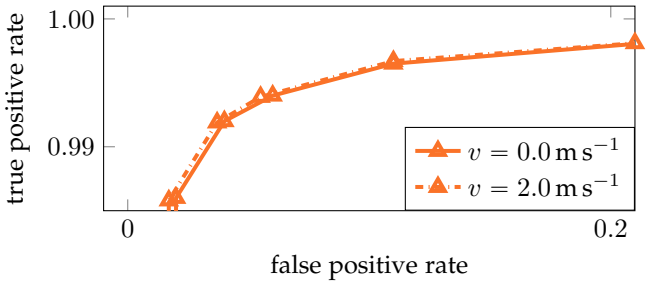
Here we discuss the difference between the ad hoc proximity test and the infrastructural proximity test. In this work, we considered a one-to-many location based handshake and private proximity test protocol. Although our design

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2015.2472529, IEEE Transactions on Dependable and Secure Computing






(**A**) WiFi location tag.




(**B**) LTE location tag.

Fig. 14. The ROC AUCs of both WiFi and LTE location tags increase slightly when users are mobile.

involved an oblivious server to ease the computational burden at the users' side, there is no need for other dedicated infrastructural support in our protocol. Most information exchanged was extracted and shared in an ad hoc, peer-to-peer fashion. The protocol can function without any certified location information. The result is that the protocol is very effective in preserving users' privacy against the service provider and other users. However, compared to other infrastructural proximity test protocols that rely on certified location information, there are several drawbacks inherent to an ad hoc proximity test.

The first one is that it is difficult to control the granularity of the location information. The granularity of our location tags is simply equal to the range of the ambient signals. We circumvent this problem by differentiating the granularity of location tags and the granularity of the proximity test. Within the coverage of location tags, we embedded a grid reference system along with an ad hoc location proof from peer users. This way, we allow the proximity test to have finer granularity than location tags. However, since both the location tags and location proofs are extracted in an ad hoc fashion, they can be unreliable under some circumstances. For instance, like we show in Fig. 9, the location proofs cannot fulfill their purpose when the population density is extremely low. The reliability of location tags also vary based the environmental factors such as spectral density *etc.*

The second drawback is due to the additional communication required by the protocol. As shown in Fig. 11, the optimal capture duration, size of the Bloom filter, etc. cannot be determined *a priori*. Therefore, Alice must share such information with the test candidates. If we wish to integrate other features into our protocol, such as negotiable grid block size, this information must also be exchanged during the protocol. To cope with such a drawback, we used mostly information broadcasting and included an oblivious server to share the computational and communication burden.

Despite the drawbacks, the security and privacy advantage of an ad hoc proximity test protocol make it ideal for privacy-aware users. Thus, we tend to consider our ad hoc proximity test a good complement to the infrastructural proximity test with non-overlapping features.

## 9 Conclusion

In this paper, we address the privacy and security issues of proximity test in LBSs. We aim at letting users find others who are within a certain geographical region or range with the help of an oblivious server, without pre-established secret keys while hiding user location information from the server. In order to prevent location cheating, we propose using multiple types of real-time and location-dependent environmental signals to construct location tags. The location tag is the key to proximity matching, where the fuzzy extractor is exploited to extract a secret key from two matching users. In addition, the location tag is organized in a Bloom filter, such that users can choose their own matching sensitivity with ease via tuning the parameter of the Bloom filter and BCH encoder. Furthermore, we also improve the accuracy and granularity of the proximity test using a geographical grid reference and keyed hashing. Through both theoretical analysis and experimental evaluation, we show that our location tag has enough freshness and entropy to defend against location cheating. Our scheme is mostly non-interactive, does not require strict synchronization, and enjoys high scalability and efficiency.

## References

[1] A. Narayanan, N. Thiagarajan, M. Lakhani, M. Hamburg, and D. Boneh, "Location privacy via private proximity testing," in *Proc. 18th Annual Network & Distributed System Security Symposium*, 2011.

[2] M. Li, N. Cao, S. Yu, and W. Lou, "Findu: Privacy-preserving personal profile matching in mobile social networks," in *Proc. 30th IEEE International Conference on Computer Communications*, 2011.

[3] M. Li, W. Lou, and K. Ren, "Data security and privacy in wireless body area networks," *Journal of Wireless Communications*, vol. 17, no. 1, 2010.

[4] X. Liang, R. Lu, L. Chen, X. Lin, and X. Shen, "Pec: A privacy-preserving emergency call scheme for mobile healthcare social networks," *Journal of Communications and Networks*, vol. 13, no. 2, 2011.

[5] W. He, X. Liu, and M. Ren, "Location cheating: A security challenge to location-based social network services," in *Proc. 31st IEEE International Conference on Distributed Computing Systems*, 2011.

[6] J. Y. Tsai, P. G. Kelley, L. F. Cranor, and N. Sadeh, "Location-sharing technologies: Privacy risks and controls," *I/S: A Journal of Law & Policy for the Information Society*, vol. 6, pp. 119–317, 2010.

[7] Z. Zhu and G. Cao, "Applaus: A privacy-preserving location proof updating system for location-based services," in *Proc. 30th IEEE International Conference on Computer Communications*, 2011.

[8] L. Šikšnys, J. R. Thomsen, S. Šaltenis, M. L. Yiu, and O. Andersen, "A location privacy aware friend locator," *Advances in Spatial and Temporal Databases*, pp. 405–410, 2009.

[9] L. Šikšnys, J. R. Thomsen, S. Šaltenis, and M. L. Yiu, "Private and flexible proximity detection in mobile social networks," in *Proc. 11th IEEE International Conference on Mobile Data Management*, 2010.

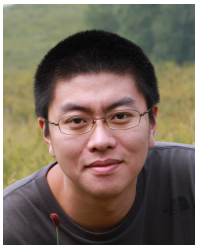
[10] Y. Dodis, R. Ostrovsky, L. Reyzin, and A. Smith, "Fuzzy extractors: How to generate strong keys from biometrics and other noisy data," in *Advances in Cryptology-Eurocrypt 2004*. Springer, 2004, pp. 523–540.

[11] J. Meyerowitz and R. R. Choudhury, "Hiding stars with fireworks: Location privacy through camouflage," in *Proc. 15th ACM Annual International Conference on Mobile Computing and Networking*, 2009.

[12] W. Chang, J. Wu, and C. C. Tan, "Enhancing mobile social network privacy," in *Proc. IEEE Global Communications Conference*, 2011.
[13] K. B. Rasmussen and S. Čapkun, "Location privacy of distance bounding protocols," in *Proceedings of the 15th ACM conference on Computer and communications security*. ACM, 2008, pp. 149–160.
[14] J.-P. Aumasson, A. Mitrokotsa, and P. Peris-Lopez, "A note on a privacy-preserving distance-bounding protocol," in *Information and Communications Security*. Springer, 2011, pp. 78–92.
[15] A. Mitrokotsa, C. Onete, and S. Vaudenay, "Location leakage in distance bounding: Why location privacy does not work," *Computers & Security*, vol. 45, pp. 199–209, 2014.
[16] S. Mascetti, C. Bettini, D. Freni, X. S. Wang, and S. Jajodia, "Privacy-aware proximity based services," in *Proc. 10th IEEE International Conference on Mobile Data Management: Systems, Services and Middleware*, 2009.
[17] Z. Lin, D. F. Kune, and N. Hoppe, "Efficient private proximity testing with gsm location sketches," in *Proc. 32nd International Cryptology Conference*, 2012.
[18] A. Z. Broder, S. C. Glassman, M. S. Manasse, and G. Zweig, "Syntactic clustering of the web," *Computer Networks and ISDN Systems*, vol. 29, no. 8-13, pp. 1157–1166, 1997.
[19] Y. Zheng, M. Li, W. Lou, and Y. T. Hou, "Sharp: Private proximity test and secure handshake with cheat-proof location tags," in *Computer Security–ESORICS 2012*. Springer, 2012, pp. 361–378.
[20] W. K. Wong, D. W. Cheung, B. Kao, and N. Mamoulis, "Secure knn computation on encrypted databases," in *Proc. 35th ACM SIGMOD International Conference on Management of Data*, 2009.
[21] M. J. Freedman, K. Nissim, and B. Pinkas, "Efficient private matching and set intersection," in *Advances in Cryptology-EUROCRYPT 2004*. Springer, 2004, pp. 1–19.
[22] L. Vu, Q. Do, and K. Nahrstedt, "Jyotish: A novel framework for constructing predictive model of people movement from joint wifi/bluetooth trace," in *Pervasive Computing and Communications (PerCom), 2011 IEEE International Conference on*. IEEE, 2011, pp. 54–62.
[23] A. Gupta, M. Miettinen, N. Asokan, and M. Nagy, "Intuitive security policy configuration in mobile devices using context profiling," in *Privacy, Security, Risk and Trust (PASSAT), 2012 International Conference on and 2012 International Confernece on Social Computing (SocialCom)*. IEEE, 2012, pp. 471–480.
[24] D. Koller and N. Friedman, *Probabilistic graphical models: principles and techniques*. MIT press, 2009.
[25] J. Franklin, D. McCoy, P. Tabriz, V. Neagoe, J. V. Randwyk, and D. Sicker, "Passive data link layer 802.11 wireless device driver fingerprinting," in *Proc. 15th USENIX Security Symposium*, 2006.
[26] M. E. Newman, "Modularity and community structure in networks," *Proceedings of the National Academy of Sciences*, vol. 103, no. 23, pp. 8577–8582, 2006.
[27] D. Qiu, D. Boneh, S. Lo, and P. Enge, "Robust location tag generation from noisy location data for security applications," in *The Institute of Navigation International Technical Meeting*, 2009.
[28] H. Yu, M. Kaminsky, P. B. Gibbons, and A. Flaxman, "Sybilguard: defending against sybil attacks via social networks," in *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 4. ACM, 2006, pp. 267–278.
[29] S. Saroiu and A. Wolman, "Enabling new mobile applications with location proofs," in *Proceedings of the 10th workshop on Mobile Computing Systems and Applications*. ACM, 2009, p. 3.
[30] E. Berlekamp, "Nonbinary bch decoding (abstr.)," *Information Theory, IEEE Transactions on*, vol. 14, no. 2, pp. 242–242, March 1968.
[31] Remcom, "Wireless InSite," Available: http://www.remcom.com/wireless-insite, 2014.
[32] T. Henderson, M. Lacage, G. Riley, M. Watrous, G. Carneiro, T. Pecorella *et al.*, "ns-3," Available: http://www.nsnam.org, 2014.
[33] Riverbed Technology Inc., "Riverbed AirPcap," Available: http://www.riverbed.com, 2014.
[34] Sanjole Inc., "IntelliJudge," Available: http://www.sanjole.com, 2014.
[35] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Commun. ACM*, vol. 13, no. 7, pp. 422–426, Jul. 1970.

## NOMENCLATURE

| | |
|---|---|
| $\mathcal{X}$ | A set of environmental signals |
| $\mathcal{Y}$ | A location tag is a subset of $\mathcal{X}$ |
| $y$ | An observation of a location tag is a element of $\mathcal{Y}$ |
| $\mathcal{Z}$ | Location features are disjoint subsets of $\mathcal{Y}$ |
| $z$ | An observation of a location feature is a element of $\mathcal{Z}$ |
| $w$ | A Bloom filter is a n-bit vector |
| $c$ | A BCH codeword is a n-bit vector |
| $s$ | A fuzzy extractor helper string is the $xor$ of $c$ and $w$ |



**Yao Zheng** is a Ph.D. student at Virginia Tech. He received his B.S. degree in Microelectronic from Fudan University in 2007, M.S. degree in Electrical Engineering from Worcester Polytechnic Institute in 2011. Between 2007 and 2009, he was a Researcher at Siemens AG, Beijing, China.

**Ming Li** is an Associate Professor in the Department of Electrical and Computer Engineering at the University of Arizona. Prior to joining the University of Arizona, he was an Assistant professor in the Department of Computer Science at Utah State University from 2011 to 2015. He received a Ph.D. in Electrical and Computer Engineering from Worcester Polytechnic Institute in 2011. His research interests are in the general areas of cyber security and wireless networks, with current emphases on security and privacy in cloud computing and big data, wireless security, and cyber-physical system security. He is a recipient of the NSF CAREER Award in 2014. He is a member of both IEEE and ACM.

**Wenjing Lou** is a professor in the Computer Science department at Virginia Polytechnic Institute and State University. Prior to joining Virginia Tech in 2011, she was a faculty member in the Department of Electrical and Computer Engineering at Worcester Polytechnic Institute (WPI) from 2003 to 2011. She received a Ph.D. in Electrical and Computer Engineering from University of Florida, an M.A.Sc. in Computer Communications from Nanyang Technological University in Singapore, and an M.S. and a B.S. in Computer Science and Engineering from Xi'an Jiaotong University in China. She worked as a Research Engineer in Network Technology Research Center at Nanyang Technological University, Singapore from 1997 to 1999.

**Y. Thomas Hou** (F'14) is the Bradley Distinguished Professor of Electrical and Computer Engineering at Virginia Tech, Blacksburg, VA, USA. He received his Ph.D. degree from NYU Polytechnic School of Engineering (formerly Polytechnic Univ.) in 1998. His current research focuses on developing innovative solutions to complex cross- layer optimization problems in wireless and mobile networks. He has published two graduate textbooks: *Applied Optimization Methods for Wireless Networks* (Cambridge University Press, 2014) and *Cognitive Radio Communications and Networks: Principles and Practices* (Academic Press/Elsevier, 2009). He is the Steering Committee Chair of the IEEE INFOCOM conference and a Distinguished Lecturer of the IEEE Communications Society.