Open access • Journal Article • DOI:10.1109/TMC.2011.237

Toward Privacy Preserving and Collusion Resistance in a Location Proof Updating System — Source link

Zhichao Zhu, Guohong Cao

Institutions: Pennsylvania State University

Published on: 01 Jan 2013 - IEEE Transactions on Mobile Computing (IEEE)

Topics: Location-based service, Information privacy, Mobile computing, Mobile device and Mobile telephony

Toward Privacy Preserving and Collusion Resistance in a Location Proof Updating System

Zhichao Zhu, *Student Member*, *IEEE*, and Guohong Cao, *Fellow*, *IEEE*

Abstract—Today's location-sensitive service relies on user's mobile device to determine the current location. This allows malicious users to access a restricted resource or provide bogus alibis by cheating on their locations. To address this issue, we propose A Privacy-Preserving LocAtion proof Updating System (APPLAUS) in which colocated Bluetooth enabled mobile devices mutually generate location proofs and send updates to a location proof server. Periodically changed pseudonyms are used by the mobile devices to protect source location privacy from each other, and from the untrusted location proof server. We also develop user-centric location privacy model in which individual users evaluate their location privacy levels and decide whether and when to accept the location proof requests. In order to defend against colluding attacks, we also present betweenness ranking-based and correlation clustering-based approaches for outlier detection. APPLAUS can be implemented with existing network infrastructure, and can be easily deployed in Bluetooth enabled mobile devices with little computation or power cost. Extensive experimental results show that APPLAUS can effectively provide location proofs, significantly preserve the source location privacy, and effectively detect colluding attacks.

Index Terms—Location-based service, location proof, location privacy, pseudonym, colluding attacks

1 Introduction

Location-based services take advantage of user location information and provide mobile users with various resources and services. Nowadays, more and more location-based applications and services require users to provide location proofs at a particular time. For example, "Google Latitude" and "Loopt" are two services that enable users to track their friends' locations in real time. These applications are location-sensitive since location proof plays a critical role in enabling these applications.

There are many kinds of location-sensitive applications. One category is location-based access control. For example, a hospital may allow patient information access only when doctors or nurses can prove that they are in a particular room of the hospital [19]. Another class of location-sensitive applications require users to provide past location proofs [26], such as *auto insurance quote* in which auto insurance companies offer discounts to drivers who can prove that they take safe routes during their daily commutes, *police investigations* in which detectives are interested in finding out if a person was at a murder scene at some time, and *location-based social networking* in which a user can ask for a location proof from the service requester and accepts the request only if the sender is able to present a valid location proof. The common theme across these location sensitive applications is that they offer a reward or benefit to users located in a certain geographical location at a certain time. Thus, users have the incentive to cheat on their locations.

Location-sensitive applications require users to prove that they really are (or were) at the claimed locations. Although most mobile users have devices capable of discovering their locations, some users may cheat on their locations and there is a lack of secure mechanism to provide their current or past locations to applications and services. One possible solution [15] is to build a trusted computing module on each mobile device to make sure trusted GPS data is generated and transmitted. For example, Lenders et al. [15] proposed such a solution which can be used to generate unforgeable geotags for mobile content such as photos and video; however, it relies on the expensive trusted computing module on mobile devices to generate proofs. Although cellular service providers have tracking services that can help verify the locations of mobile users in real time, the accuracy is not good enough and the location history can not be verified. Recently, several systems have been designed to let end users prove their locations through WiFi infrastructures. For example, Saroiu and Wolman [26] proposed a solution suitable for third-party attestation, but it relies on PKI and the wide deployment of WiFi infrastructure.

In this paper, we propose A Privacy-Preserving LocAtion proof Updating System (APPLAUS), which does not rely on the wide deployment of network infrastructure or the expensive trusted computing module. In APPLAUS, Bluetooth enabled mobile devices in range mutually generate location proofs, which are uploaded to a untrusted location proof server that can verify the trust level of each location proof. An authorized verifier can query and retrieve location proofs from the server. Moreover, our location proof system guarantees user location privacy from every party. More specifically, we use statistically updated pseudonyms at each mobile device to protect

- *The authors are with the Department of Computer Science and Engineering, The Pennsylvania State University, Information Science and Technology Building, University Park, PA 16802. E-mail: {zzhu, gcao}@cse.psu.edu.*

Manuscript received 25 Apr. 2011; revised 16 Aug. 2011; accepted 12 Oct. 2011; published online 4 Nov. 2011.
For information on obtaining reprints of this article, please send e-mail to: tmc@computer.org, and reference IEEECS Log Number TMC-2011-04-0214.
Digital Object Identifier no. 10.1109/TMC.2011.237.

1536-1233/13/$31.00 © 2013 IEEE Published by the IEEE CS, CASS, ComSoc, IES, & SPS

location privacy from each other, and from the untrusted location proof server. We develop a user-centric location privacy model in which individual users evaluate their location privacy levels in real time and decide whether and when to accept a location proof request. In order to defend against colluding attacks, we also present betweenness ranking-based and correlation clustering-based approaches for outlier detection. Extensive experimental and simulation results based on multiple data sets show that APPLAUS can effectively provide location proofs, significantly preserve the source location privacy, and effectively detect colluding attacks.

The rest of the paper is organized as follows: We first introduce preliminaries of our scheme in Section 2, and then present our location proof updating scheme in Section 3. Section 4 presents the source location privacy analysis and Section 5 discusses colluding attacks and countermensures. The performance of our scheme is evaluated in Section 6. Finally, we describe related work in Section 7 and conclude the paper in Section 8.

2 Preliminaries

In this paper, we focus on mobile networks where mobile devices such as cellular phones communicate with each other through Bluetooth. In our implementation, mobile devices periodically initiate location proof requests to all neighboring devices through Bluetooth. After receiving a request, a mobile node decides whether to exchange location proof, based on its own location proof updating requirement and its own privacy consideration. Given its appropriate range (about 10 m) and low power consumption, Bluetooth is a natural choice for mutual encounters and location proof exchange.

2.1 Pseudonym

As commonly used in many networks, we consider an online Certification Authority (CA) run by independent trusted third party which can preestablish credentials for the mobile devices. Similar to many pseudonym approaches, to protect location privacy, every mobile node i registers with the CA by preloading a set of M public/private key pairs $K_i^{Pub}, K_i^{Prv}{}_{i=1}^{M}$ before entering the network. The public key K_i^{Pub} is used to serve as the pseudonym of node i. The private key K_i^{Prv} enables node i to digitally sign messages so that the receiver can validate the signature authenticity.

Due to the broadcast nature of wireless communication, probes are used for mobile nodes to discover their neighbors. When a node i receives a probe from another node, it checks the certificate of the public key of the sender and the physical identity, e.g., Bluetooth MAC address. After that, i verifies the signature of the probe message. Subsequently, if confidentiality is required, a security association is established (e.g., with Diffie-Hellman).

2.2 Threat Model

We assume that an adversary aims to track the location of a mobile node. An adversary can have the same credential as a mobile node and is equipped to eavesdrop communications. We assume that the adversary is internal, passive, and global. By internal, we mean that the adversary is able to compromise or control individual mobile device and then communicate with others to explore private information, or individual devices may collude with each other to generate false proofs, which will be discussed in detail in Section 5. We assume that the number of colluders is small compared with that of valid devices. In the worst case, the adversary could compromise the location proof server to get the stored location proof records. However, it is not able to take control of the server to work as a colluder, since once compromised, the attack will be detected promptly and the location proof server will be replaced by a back-up server. The same assumption applies to the CA. By passive, we assume the adversary cannot perform active channel jamming, mobile worm attacks [34] or other denial-of-service attacks, since these attacks are not related to location privacy. By global, we assume the adversary can monitor, eavesdrop, and analyze all the traffic in its neighboring area, or even monitor all the traffic around the server.

In practice, the adversary can thus be a rogue individual, a set of malicious mobile nodes, or eavesdropping devices in the network. In the worst case, it is possible that the untrusted location proof server may be compromised by the adversary and the location information can then be easily inferred by examining the records of location proofs, e.g., the adversary could apply statistical testing such as K-S test to identify a user although no real identity is included. Therefore, we need to appropriately design and arrange the location proof records in the untrusted server so that no private information related to individual users will be revealed even after it is compromised. Hence, the problem we address in this paper consists of collecting a set of location proofs for each peer node and protecting the location privacy of peer nodes from each other, from the adversary, or even from the untrusted location proof server to prevent other parties from learning a node's past and current location information.

2.3 Location Privacy Level

In this paper, we use multiple pseudonyms to preserve location privacy; i.e., mobile nodes periodically change the pseudonym used to sign messages, thus reducing their long term linkability. To avoid spatial correlation of their location, mobile nodes in proximity coordinate pseudonym changes by using silent mix zones [16], [17], or regions where the adversary has no coverage [4]. Without loss of generality, we assume each node changes its pseudonyms from time to time according to its privacy requirement. If this node changes its pseudonym at least once during a time period (mix zone), a mix of its identity and location occurs, and the mix zone becomes a confusion point for the adversary.

Consider a mobile network composed of N mobile nodes and each node has M pseudonyms. At time t, for each node i there are a group of $m(t)$ pseudonyms observed at the location proof server. Each pseudonym among the $m(t)$ pseudonyms can involve multiple location proofs across various locations $l_1, l_2, \ldots, l_n$ at different time $t_1, t_2, \ldots, t_n$. An adversary is able to correlate the location and time distribution of each pseudonym to see if two pseudonyms belong to the same node. For example, the adversary can observe a series of location proofs with $m(T)$ pseudonyms during time T. He then compares the distribution of location proof set B of pseudonym b with the distribution of location proof set D of pseudonym d to determine if the two pseudonyms can be linked. Let $p_{d=b} = \Pr$ (distribution D of pseudonym corresponds to distribution B of pseudonym b),

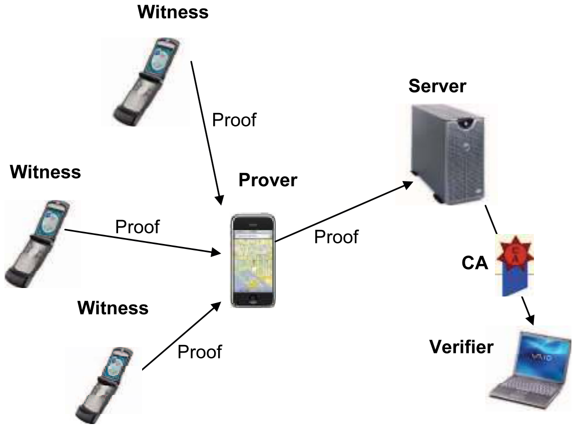


Fig. 1. Location proof updating architecture and message flow.

the location privacy level of node i (i.e., the uncertainty of the adversary) at time T, is

$$E_i(T) = -\sum_{d,b \in m(T)} p_{d=b} log_2(p_{d=b}). \tag{1}$$

The achievable location privacy depends on the number of nodes ($m(T)$) and the unpredictability of their whereabouts in the mix zone ($p_{d|b}$). If a node i has only one pseudonym observed till time T, its identity is known to the adversary and its location privacy level is defined to be $E_i(T) = 0$. We can achieve the maximum entropy when every $p_{d=b}$ is close to 0; i.e., the distribution of location proofs for each pseudonym is undistinguishable.

3 The Location Proof Updating System

In this section, we introduce the location proof updating architecture, the protocol, and how mobile nodes schedule their location proof updating to achieve location privacy in APPLAUS.

3.1 Architecture

In APPLAUS, mobile nodes communicate with neighboring nodes through Bluetooth, and communicate with the untrusted server through the cellular network interface. Based on different roles they play in the process of location proof updating, they are categorized as Prover, Witness, Location Proof Server, Certificate Authority or Verifier. The architecture and message flow of APPLAUS is shown in Fig. 1.

- **Prover**: the node who needs to collect location proofs from its neighboring nodes. When a location proof is needed at time t, the prover will broadcast a location proof request to its neighboring nodes through Bluetooth. If no positive response is received, the prover will generate a dummy location proof and submit it to the location proof server.
- **Witness**: Once a neighboring node agrees to provide location proof for the prover, this node becomes a witness of the prover. The witness node will generate a location proof and send it back to the prover.
- **Location proof server**: As our goal is not only to monitor real-time locations, but also to retrieve history location proof information when needed, a location proof server is necessary for storing the history records of the location proofs. It communicates directly with the prover nodes who submit their location proofs. As the source identities of the location proofs are stored as pseudonyms, the location proof server is untrusted in the sense that even though it is compromised and monitored by attackers, it is impossible for the attacker to reveal the real source of the location proof.
- **Certificate authority:** As commonly used in many networks, we consider an online CA which is run by an independent trusted third party. Every mobile node registers with the CA and pre-loads a set of public/private key pairs before entering the network. CA is the only party who knows the mapping between the real identity and pseudonyms (public keys), and works as a bridge between the verifier and the location proof server. It can retrieve location proof from the server and forward it to the verifier.
- **Verifier**: a third-party user or an application who is authorized to verify a prover's location within a specific time period. The verifier usually has close relationship with the prover, e.g., friends or colleagues, to be trusted enough to gain authorization.

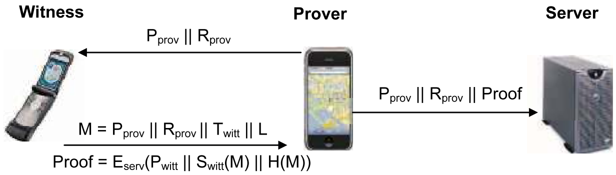

Witness
Prover
Server
Pprov || Rprov
Pprov || Rprov || Proof
M = Pprov || Rprov || Twitt || L
Proof = Eserv(Pwitt || Switt(M) || H(M))

Fig. 2. Location proof updating protocol.

3.2 Protocol

When a prover needs to collect location proofs at time t, it executes the protocol in Fig. 2 to obtain location proofs from the neighboring nodes within its Bluetooth communication range. Each node uses its M pseudonyms $P_{i=1}^{M}$ as its identity throughout the communication.

1. The prover broadcasts a location proof request to its neighboring nodes through Bluetooth according to its update scheduling. The request should contain the prover's current pseudonym P_{prov}, and a random number R_{prov}.
2. The witness decides whether to accept the location proof request according to its witness scheduling. Once agreed, it will generate a location proof for both prover and itself and send the proof back to the prover. This location proof includes the prover's pseudonym P_{prov}, prover's random number R_{prov}, witness's current time stamp T_{witt}, witness's pseudonym P_{witt}, and their shared location L. This proof is signed and hashed by the witness to make sure that no attacker or prover can modify the location proof and the witness cannot deny this proof. It is also encrypted by the server's public key to prevent from traffic monitoring or eavesdropping.
3. After receiving the location proof, the prover is responsible for submitting this proof to the location proof server. The message also includes prover's pseudonym P_{prov} and random number R_{prov}, or its own location for verification purpose.

4. An authorized verifier can query the CA for location proofs of a specific prover. This query contains a real identity and a time interval. The CA first authenticates the verifier, and then converts the real identity to its corresponding pseudonyms during that time period and retrieves their location proofs from the server. In order not to expose correlation between pseudonyms to the location server, CA will always collect enough queries from k different nodes before a set of queries are sent out.
5. The location proof server only returns hashed location rather than the real location to the CA, who then forwards to the verifier. The verifier compares the hashed location with the claimed location acquired from the prover to decide if the claimed location is authentic.

In order to prevent the CA from knowing locations of a real identity, the location proof server calculates the hash of each location and only sends the hashed locations to the CA in step 5. In this way, the following property can be achieved.

Definition 1 (Separation of privacy knowledge). *The knowledge of the privacy information is separately distributed to the location proof server, the CA, and the verifier. Thus, each party only has partial knowledge.*

The privacy property of our protocol is ensured by the separation of privacy knowledge: the location proof server only knows pseudonyms and locations, the CA only knows the mapping between the real identity and its pseudonyms, while the verifier only knows the real identity and its authorized locations. Attackers are unable to learn a user's location information without integrating all the knowledge. Therefore, compromising either party of the system does not reveal privacy information.

3.3 Scheduling Location Proof Updates

As discussed before, the adversary may obtain complete coverage and track nodes throughout the entire network, by compromising the location proof server and obtain all history location proofs. Therefore, we need to appropriately design and arrange the location proof updating schedules for both prover and witness so that no source location information related to individual user is revealed even if the server is compromised.

Suppose a mobile node i has a set of pseudonyms $P_1, P_2, \ldots, P_M$ which change periodically, and distinct parameters $\lambda_1, \lambda_2, \ldots, \lambda_M$ for each pseudonym are predetermined. If each pseudonym P_j updates its location proofs such that the interupdate interval follows Poisson distribution with parameter λ_j, as shown in Fig. 3, then the entire interupdate intervals for node i follow Poisson distribution with a parameter of $\lambda = \lambda_1 + \lambda_2 + \cdots + \lambda_M$. As will be discussed in the next section, it has the properties of pseudonym unlinkability and statistically strong source location unobservability. The detailed scheduling protocol for the prover is shown in Algorithm 1. The predefined updating parameter λ determines how frequently location proofs are updated. In some cases, no location proof is generated when the location proof updating time arrives. To ensure that location proof updating follows the scheduled Poisson distribution, a dummy proof is generated and submitted. The dummy proof has the same format as the real location proof and cannot be differentiated by the attackers.

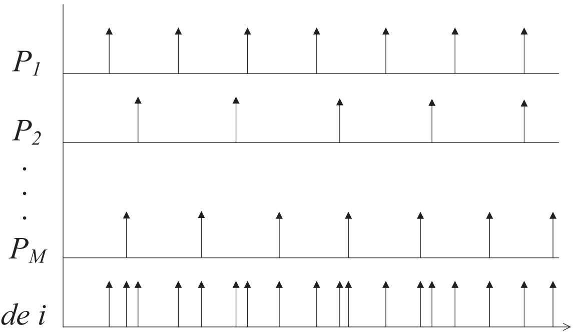


Fig. 3. Example of individual pseudonym update versus entire node update.

Algorithm 1. Location Proof Update Scheduling for the prover

```
Input: updating parameter $\lambda$;
generate $M$ distinct parameter $\lambda_1, \lambda_2, \cdots, \lambda_M$ such
  that $\lambda_1 + \lambda_2 + \cdots + \lambda_M = \lambda$
for each pseudonym $i$ do
    while current timestamp $t$ follows Poisson
    distribution with $\lambda_i$ do
      send location proof request
      if request is accepted then
          submit location proof
      else
          generate and submit dummy proof
      end if
    end while
end for
```

The location privacy of witness nodes varies depending on the time and location when they exchange location proofs. It is thus desirable to protect the location privacy in a user-centric manner, such that each user can decide when and how to protect his location privacy. User-centric location privacy [12], [17] follows a distributed approach where each mobile node locally monitors its location privacy level over time. A network wide metric measures the average location privacy but may ignore that some nodes have low location privacy levels. However, the user centric approach is more scalable and can maintain location privacy at a more fine-grained level. In our model, the location privacy of a node may accumulate over time. It depends on the distribution diversity of the last pseudonym and its previous pseudonyms before the last successful pseudonym change. Each mobile node monitors and measures its own privacy level in real time and decides whether and when to accept a location proof exchange request. After receiving a location proof exchange request, it calculates the privacy loss between the next scheduled updating time and the current updating time. In this way, a node has autonomy to control the time period over which its location is tracked. The privacy loss of node i is defined as follows:

$$\Delta = \frac{E_i(t') - E_i(t)}{E_i(t)}, \tag{2}$$

where $E_i(t)$ is the location privacy level when the location proof exchange request is accepted while $E_i(t')$ is the location privacy level at the next scheduled location proof updating cycle. The difference between them indicates the privacy loss if this location proof request is accepted. The location proof request is only accepted when the privacy loss is less than a predefined threshold. The drawback of the user-centric model is that nodes may have misaligned incentives (i.e., different privacy requirement), which can lead to failed attempts to obtain enough location proofs. We use dummy proofs in Algorithm 1 to deal with failed attempts. The detailed scheduling protocol for witness is presented in Algorithm 2.

Algorithm 2. Scheduling Location Proof Updates at Witnesses

```
Input: time $t$ of incoming location proof request;
calculate location privacy loss $\Delta$ assuming the
    incoming request is accepted
if $\Delta > \epsilon$, $\epsilon$ is pre-defined location privacy loss threshold
then
    deny location proof request
else
    accept location proof request
end if
```

4 Source Location Privacy Analysis

In this section, we discuss the location privacy threat in our system, as well as our countermeasures.

We first look at how an adversary may reveal location information by analyzing the location proof history. Suppose the attacker has sufficient resources (e.g., in storage, computation and communication). First, the attacker may simply monitor and examine the content of a record that contain the user's identity and location. Second, even if the user's ID is encrypted or pseudonymized, it is easy for the adversary to trace back all the location activities related to the same ID once its pseudonym is discovered. Third, even though the user's pseudonyms change periodically, it is still possible for the adversary to infer this user's other pseudonyms from one pseudonym if these pseudonyms change at similar time or locations. Moreover, the attacker may perform more advanced traffic analysis including rate monitoring and location correlation. In a rate monitoring attack, the attacker tries to monitor and correlate location proof updating rates from different pseudonyms. In a location correlation attack, the attacker may observe the correlation in the updated location between a node and its neighbors, attempting to deduce a relationship.

According to [23], a mechanism to achieve anonymity appropriately combined with dummy traffic yields unobservability, In our case, we are interested in the privacy property of each node, and we have the following definition:

Definition 2. *Source location unobservability is a privacy property that can be satisfied if an attacker cannot determine the real identity of mobile nodes through observation of the location proof records. That is, for each possible observation O that an attacker can make, if the probability of an identity I is equal to the probability of I given O; i.e., $\forall O, P(I) = P(I|O)$, then I is called unobservable.*

Based on the above definition, we have the following definition on pseudonym unlinkability:

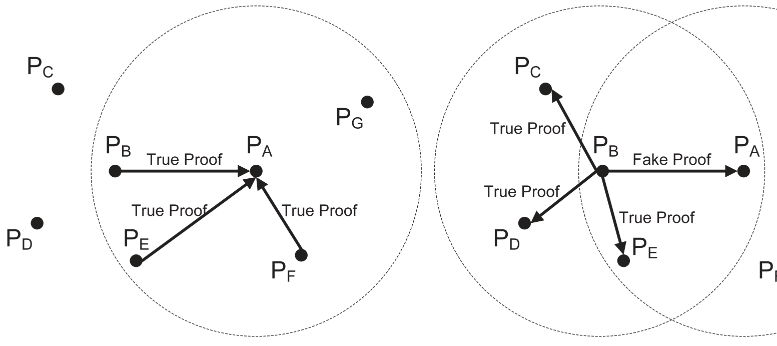
Definition 3. *A system has the property of **pseudonym unlinkability** if the pseudonyms $P_1, P_2, \ldots, P_M$ of an identity I presented in the location proof records cannot be inferred from one to another: $\forall i, j, \forall O$, prob$(P_i|O) \neq$ prob$(P_j|O)$, $i, j \in \{1, 2, \ldots, M\}, i \neq j$.*

Obviously, a system satisfies source location unobservability if and only if it has the property of pseudonym unlinkability. We need to design a probabilistic solution, which can provide a satisfactory degree of event unobservability and reduce the latency as much as possible. For example, we can adopt a Poisson distribution to determine the time intervals between location proof updates. Under our attack model, an adversary can easily know a constant rate distribution and its mean by statistic test over the history of location proof records. However, if we use a probabilistic rate scheme and keep the seed for generating random numbers secret from an attacker, the attacker may not be able to notice if a proof update is due to a real event or a dummy message even if a node sends out a real event message immediately. Intuitively, a node cannot always initiate or accept a proof request immediately in the presence of burst events; otherwise, an attacker may notice the change of the underlying distribution. Therefore, it is difficult to guarantee perfect event unobservability while providing low latency. Thus, we adopt **statistically strong source location unobservability** to achieve low latency and high privacy.

Suppose the interupdate delay (d) between location proof updates $k (k > 0)$ and $k+1$ from a pseudonym $i (1 \leq i \leq M)$ of a mobile node is $d_k^i = t_{k+1}^i - t_k^i$, where t_k^i is the update time of location proof k from pseudonym i. A global attacker can observe a sequence of interupdate delays, which can be represented as a distribution $X_i = d_1^i, d_2^i, \ldots$. Ideally, in a scheme with perfect unobservability but high latency, interupdate delays from all the pseudonyms of the same node follow different distribution, so that an attacker cannot identify the relations between these pseudonyms. In our case of statistically strong source location unobservability, distributions of interupdate delays by different pseudonyms are statistically distinct from each other. They are distinct from each other in the sense that by a statistic test one cannot correlate them with each other. We have the definition of statistically distinct distributions as follows:

Definition 4. *Two probabilistic distinct distribution X_i and $X_j (1 \leq i, j \leq M, i \neq j)$ are statistically distinct from each other if they follow the same type of probabilistic distribution with distinct parameter.*

Clearly, the more parameters a distribution has, the harder it is to prove its statistical distinction. In our scheme, we decided to use Poisson distribution to control the rate of location proof updating due to its one-parameter advantage, which makes it relatively easy to achieve source location unobservability. More specifically, we have the following lemma:



(a) Normal case

(b) Colluding case

Fig. 4. Threshold based verification. (a) all nodes are legitimate nodes. $TL_{P_A} = \frac{N_{proof}}{N_{neighbor}} = 0.75$. (b) P_A tries to claim her location in NYC with her colluders although she is not at NYC. P_B is P_A's colluder and witness. $TL_{P_A} = 1/4 = 0.25$, and P_A is detected as malicious.

Lemma 1. *Suppose a mobile node has a set of pseudonyms $P_1, P_2, \ldots, P_M$ which change periodically. Each pseudonym i sends out its location proof updates (including dummy updates) whose interupdate delay follows Poisson distribution with a parameter λ_i. Then all the interupdate delays of this mobile node follow Poisson distribution with parameter $\lambda = \lambda_1 + \lambda_2 + \cdots + \lambda_M$.*

Proof 1. Without loss of generality, we assume the mobile node changes its pseudonyms in the order of $P_1, P_2, \ldots, P_M$. As each pseudonym i follows Poisson distribution $X_i = P(X = t) = \frac{\lambda_i^t e^{-\lambda_i}}{t!}$, the distribution of the combination of pseudonyms P_i and P_j is

$$Y = P(X_i + X_j = k) \tag{3}$$

$$= \sum_{t=0}^{k} P(X_i = t) P(X_j = k - t) \tag{4}$$

$$= \sum_{t=0}^{k} \frac{\lambda_i^t e^{-\lambda_i}}{t!} \cdot \frac{\lambda_j^{(}k - t) e^{-\lambda_i}}{(k - t)!} \tag{5}$$

$$= \frac{(\lambda_i + \lambda_j)^t e^{-(\lambda_i + \lambda_j)}}{t!}. \tag{6}$$

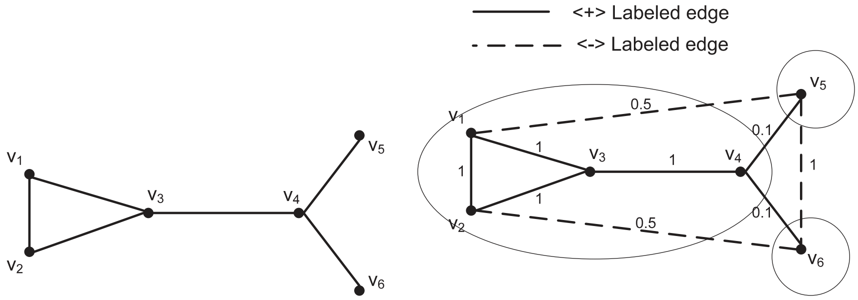
That is, the sum of two independent Poisson distributions with parameters of λ_i and λ_j also follow a Poisson distribution with parameter $\lambda_i + \lambda_j$. It is easy to extend that the sum of all pseudonyms also follows Poisson distribution with a parameter of $\lambda = \lambda_i + \lambda_2 + \cdots + \lambda_M$. □

Therefore, if each mobile node in the network chooses M distinct parameters from λ_1 to λ_M for its M pseudonyms, and schedules location proof updates based on the aforementioned Poisson distributions, the location proof records in the server have the properties of pseudonym unlinkability and statistically strong source location unobservability. When parameter λ is fixed, increasing the number of pseudonyms (M) improves the privacy level, as it increases the possibility for attackers to confuse between pseudonyms from different nodes. Meanwhile, as each node presents more pseudonyms in the system, it also decreases the possibility for this node to be identified even though one of its pseudonym has been recognized. Quantifying the relationship between the number of pseudonyms (M) and the privacy level is difficult since the privacy level is determined by many other parameters. However, the upper bound of M may always be determined due to the space limit of mobile devices. There are a number of ways to pick individual λ_i when λ is fixed. For example, one intuitive solution is to generate an arithmetic progression or a geometric progression which sums to λ.

5 Colluding Attacks and Countermeasures

The joint issues of location proof and location privacy have been studied in [33], but the threat of colluding attack is still an open issue. This threat exists when two nodes collude with each other to generate bogus location proofs. For example, when a dishonest node C_1 from San Francisco needs to prove herself in New York City (NYC), she can have another colluding node C_2 to generate bogus location proofs for her, with location tag of New York City. Generally speaking, such attacks can be identified by looking into the location traces and examining the interactions between colluders as well as the time and location consistency along the moving trajectory. We first consider statistical threshold based solution in which the system requires the prover to obtain a number of witness nodes, no matter what their real identities are. As we know, the location proof server has information about the number of pseudonyms at a particular time and location. This information can be used to estimate whether the prover lies about not finding enough peers or always finding the same peer based on some statistical techniques.

More specifically, the server-level detection is performed on individual location proof based on its embedded time stamp and location information, where all concurrent and co-located location proofs from other nodes (pseudonyms) are used to verify its trust level. For example in the normal case, as in Fig. 4a, P_A is a legitimate prover who has received location proofs from all neighboring witness nodes P_B, P_E, and P_F, except P_G due to interference or synchronization reasons. Therefore, P_A has collected three location proofs ($N_{proof} = 3$) from four neighbors ($N_{neighbor} = 4$), and we denote the *trust level* of this location proof as $TL_{P_A} = \frac{N_{proof}}{N_{neighbor}} = 0.75$. Since the trust level TL is



(a) Unweighted pseudonym-correlation graph (b) Weighted proof-correlation graph and correlation clustering

Fig. 5. Examples of unweighted pseudonym-correlation graph and temporal-weighted proof-correlation graph.

higher than preselected threshold, P_A is considered as a good node.

Fig. 4b illustrate a colluding case where node P_A tries to claim her location in New York city with her colluders although she is not at NYC. Although P_A finds a colluder P_B who is located in NYC to generate a fake location proof for her, it can't use other provers since she is not at NYC. The location proof server looks through the location proof records to check if there are other provers in P_A's communication range. In this example, several other nodes (e.g., P_E, P_F, and P_G) exist. It is easy to calculate the trust level of P_A's location proof is $TL_{P_A} = 1/4 = 0.25$, which is below the threshold. Thus, P_A is listed as suspicious to be malicious.

Calculating the trust level of a location proof involves the examination of its surrounding location proofs for both prover and witness, as well as large amount of redundant calculations between individual location proofs. To address this problem, we develop techniques that can perform verifications on a set of location proofs which are relevant in time and space, rather than individual proofs. We present two approaches to detect suspicious location proofs and pseudonyms: *betweenness ranking* and *correlation clustering*. The *betweenness ranking* approach calculates the betweenness of each pseudonym in a graph and then ranks these pseudonyms based on their betweenness value. The pseudonyms with low ranking are considered as suspicious nodes. The *correlation clustering* approach takes into account the time delay between two neighboring location proofs, and uses a modified correlation clustering algorithm on a temporal-weighted graph to rule out outlier clusters, which are considered as suspicious location proofs. Both approaches use undirected graph to reflect the relationship between pseudonyms or between location proofs.

5.1 Betweenness Ranking

As the pseudonyms of each node keep changing, we don't have the contact patterns of individual nodes except their pseudonyms. Therefore, we can only verify pseudonyms that are relevant enough in both temporal and spatial domains. Considering all the concurrent (within 60 seconds delay) location proofs in a region, we first describe how to construct a pseudonym-correlation graph.

Definition 5. *A pseudonym-correlation graph for region R is an undirected graph $G = (V, E)$, with vertex set $v_1, v_2, \ldots, v_n$, were $n = |V|$. Each vertex $v_i \in V$ represents a pseudonym P_i in region R, while each edge $(v_i, v_j) \in E$ denotes that there is a location proof generated between node v_i and node v_j.*

Let $s = v_0$ and $t = v_l$. A path from s to t is defined as a sequence of edges (v_i, v_{i+1}), $0 \leq i \leq l$. The *length* of a path is the number of edges in the sequence. We use $d(s,t)$ to denote the *distance* (the minimum length of any path connecting s and t in G) between s and t. With $d(i,i) = 0$, we have the following definition of betweenness:

Definition 6. *Let σ_{st} (which is equal to σ_{ts}) denote the total number of shortest paths from s to t, where $\sigma_{ss} = 1$. Let $\sigma_{st}(v)$ denote the number of shortest paths from s to t which include node $v \in V$. Then the betweenness of v is as following:*

$$B(v) = \sum_{s \neq v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}}. \tag{7}$$

Betweenness is defined as the number of shortest paths from all vertices to all others that pass through node v. It is an indicator of who is the most influential node in the network (i.e., who interconnects with most others). As in Fig. 5a, there are seven shortest paths ($v_1 - v_5, v_1 - v_6, v_2 - v_5, v_2 - v_6, v_3 - v_5, v_3 - v_6, v_5 - v_6$) pass through v_4, the betweenness of node v_4 can be calculated as $B(v_4) = \sum_7 (1) = 7$. Similarly, we can get $B(v_5) = B(v_6) = 0$. Obviously v_5 and v_6 have higher chance to be malicious node than v_4. One notable feature of the definition is that the betweenness measurement gives very clear winners and losers among the nodes in the network: individuals with the highest betweenness are well ahead of those with the second highest, who are in turn well ahead of those with the third highest, and so on. Individuals with the lowest betweenness usually only connect to one or few neighbors. They are treated as outliers among all the nodes and thus most likely are colluding attackers.

Let $\delta_{st}(v) = \frac{\sigma_{st}(v)}{\sigma_{st}}$. The *dependency* of a source vetex s on a vertex v is defined as

$$\delta_{s*}(v) = \sum_{t: t \neq s, t \neq v} \delta_{st}(v). \tag{8}$$

The betweenness of a vertex v can be expressed as

$$B(v) = \sum_{s \neq v} \delta_{s*}(v). \tag{9}$$

Define the set of *predecessors* of a vertex v on the shortest paths from s as $P_s(v) = u \in V\colon (u,v) \in E, d(s,v) = d(s,u) + 1$.

The following theorem states that the dependencies of the closer vertices can be computed from the dependencies of the farther vertices.

Lemma 2. *If there is exactly one shortest path from s to each t, the dependency of s on any vertex v obeys*

$$\delta_{s*}(v) = \sum_{w: v \in P_s(w)} (1 + \delta_{s*}(w)). \tag{10}$$

Proof 2. The assumption implies that the vertices and edges of all shortest paths from s form a tree. Therefore, v lies on either all or none of the paths between s and some vertex t, i.e., $\delta_{st}(v)$ equals either 1 or 0. Moreover, v lies on all shortest paths to those vertices for which it is a predecessor, and on all shortest paths that these lie on.□

In the general case, a very similar theorem [2] holds below:

Theorem 1. *The dependency of s on any vertex v obeys*

$$\delta_{s*}(v) = \sum_{w: v \in P_s(w)} \frac{\sigma_{sv}}{\sigma_{sw}} \cdot (1 + \delta_{s*}(w)). \tag{11}$$

The naive way to calculate betweenness takes time of order $O(mn^2)$, since there are $O(n^2)$ shortest paths to be considered, each of which takes $O(m)$ to calculate. However, since breadth-first search algorithms can calculate n shortest paths in time $O(m)$, we can calculate betweenness for all vertices in time $O(mn)$. Here, we present an algorithm that performs betweenness calculation based on the above theorem.

In Algorithm 3, n single-source shortest paths are first computed, for each s. The predecessor sets $P_s(w)$ are maintained during these computations. Next, for every s, using the information from the shortest path tree and predecessor sets along the paths, we can compute the dependencies $\delta_{s*}(v)$ for all other vertex v. To obtain the betweenness value of vertex v, we then calculate the sum of all dependency values. The $O(n^2)$ space requirements can be reduced to $O(n + m)$ by maintaining a running betweenness score. This algorithm runs in $O(nm)$ time for unweighted pseudonym-correlation graphs.

Algorithm 3. Betweenness calculation on pseudonym-correlation graph

```
Input: $B[v] = 0$, $v \in V$; $S \leftarrow$ empty stack; $Q \leftarrow$ empty queue;
for $s \in V$ do
    $P[w] \leftarrow$ empty list, $\sigma[w] = 0$, $d[w] = -1$, $w \in V$;
    $\sigma[s] = 1$; $d[s] = 0$;
    enqueue $s \rightarrow Q$
    while $Q$ not empty do
        dequeue $v \leftarrow Q$;
        push $v \rightarrow S$;
        for each neighbor $w$ of $v$ do
            if $w$ found for the first time then
                enqueue $w \rightarrow Q$;
                $d[w] = d[v] + 1$;
            end if
            if $d[w] = d[v] + 1$ then
                $\sigma[w] = \sigma[w] + \sigma[v]$
                append $v \rightarrow P[w]$;
            end if
        end for
    end while
    $\delta[v] = 0$, $v \in V$;
    while $S$ not empty do
        pop $w \leftarrow S$;
        for $v \in P[w]$ do
            $\delta[v] = \delta[v] + \frac{\sigma[v]}{\sigma[w]} \cdot (1 + \delta[w])$;
        end for
        if $w \neq s$ then
            $B[w] = B[w] + \delta[w]$;
        end if
    end while
end for
```

5.2 Correlation Clustering

The above approach only considers two location proofs correlated if they occurred at the same time (e.g., within 60 seconds). However, in most case, a node may have to wait for a time period until its next location proof updating cycle. If the time delay between two location proofs is not too large, we should still consider them correlated. Therefore, if we consider each location proof as a vertex, we have the following definition of temporal-weighted proof-correlation graph.

Definition 7. *A weighted proof-correlation graph for region R is an undirected graph $G = (V, E)$, with vertex set $v_1, v_2, \ldots, v_n$, where $n = |V|$. Each vertex $v_i \in V$ represents a location proof LP_i in region R. Each positive edge $(v_i, v_j) \in E$ denotes that location proofs LP_i and LP_j share prover or witness while each negative edge $(v_i, v_j) \in E$ denotes that LP_i's prover is actually LP_j's witness, or vice versa. For each edge (i, j), let t_{ij} be the time difference between two location proofs represented by v_i and v_j, where $0 \leq t_{ij} < +\infty$, then we define weight function c_{ij} for each edge (i, j) as follows:*

$$c_{ij} = \frac{1}{1 + t_{ij}}. \tag{12}$$

The weight function c_{ij} has both $0 \leq c_{ij} \leq 1$ for positive and negative edges. The smaller the time interval t_{ij} is, the larger the weight c_{ij} is.

Fig. 5b shows one example of weighted proof-correlation graph. Location proof v_1 and location proof v_2 share the same prover node, so the edge between them is positive with weight of 1. On the other hand, location proof v_5's prover is also location proof v_6's witness, so the edge is negative. Our goal is to have location proofs with position edges grouped together (e.g., v_1 and v_2), and to separate the location proofs with negative edges (e.g., v_5 and v_6). The reason is intuitive: the location proofs with negative edges are abnormal, since within similar time interval and similar space, there should be another location proof consisting of v_5's prover and v_6's witness. Fig. 5b shows how correlation clustering works on the graph. Since this problem is NP-hard, we transfer the above proof-correlation graph to a double-labeled graph.

With $G=(V,E)$, each edge (i,j) has a weight function c_{ij}. Let $e(u,v)$ denote the label $(\langle+\rangle, \langle-\rangle)$ of the edge (u,v). Let $E^{\langle+\rangle}$ be the set of positive edges (where $0 \leq t_{ij} \leq 1$) and let $G^{\langle+\rangle}$ be the graph induced by $E^{\langle+\rangle}$, $G^{\langle+\rangle} = (V, E^{\langle+\rangle})$. We then define $E^{\langle-\rangle}$ and $G^{\langle-\rangle}$ for negative edges in the same way. In our scenario, the weight value from one node to another can be treated as a measure of similarity, with high weight indicating similarity and low weight indicating dissimilarity. We can perform a correlation clustering over the graph, grouping nodes together who have higher positive weight with others, and separating nodes who have higher negative weight with others. Since the time complexity is too high for correlation clustering, we focus on approximation algorithms to achieve better performance.

Let OPT represent the optimal clustering algorithm, in which positive labeled edges are always in one cluster, while negative labeled edges are always between different clusters. Our goal is to propose an approximation algorithm close to OPT, by minimizing the weight of positive edges between clusters and the weight of negative edges inside clusters. Also, the approximation algorithm should maximize the weight of positive edges inside clusters and maximize the weight of negative edges between clusters. Given a clustering algorithm S, the difference between S and OPT is denoted as $d(S)$ which is the sum of the weights of negative labeled edges inside a cluster, plus weights of positive labeled edges between clusters in S. We then formulate the problem with linear programming and then give an approximation algorithm. Consider assigning a variable x_{ij} to each pair of vertices ($x_{ij} = x_{ji}$); $x_{ij} = 0$ if v_i and v_j are in a common cluster, and $x_{ij} = 1$ otherwise. As $1 - x_{ij}$ is 1 if edge (i,j) is within a cluster and 0 if edge (i,j) is between clusters, we have the following:

$$d(S) = \sum_{(i,j)\in E^{\langle-\rangle}} c_{ij}(1 - x_{ij}) + \sum_{(i,j)\in E^{\langle+\rangle}} c_{ij}x_{ij}. \tag{13}$$

Our goal is to find an assignment of x_{ij} to minimize the difference $d(S)$. We relax the requirement and get the following linear program:

$$\text{minimize} \sum_{(i,j)\in E^{\langle-\rangle}} c_{ij}(1 - x_{ij}) + \sum_{(i,j)\in E^{\langle+\rangle}} c_{ij}x_{ij}, \tag{14}$$

$$\text{subject to } x_{ij} \in [0,1], \tag{15}$$

$$x_{ij} + x_{jk} \geq x_{ik}, \tag{16}$$

$$x_{ij} = x_{ji}. \tag{17}$$

We present a $O(logn)$ approximation solution to this linear program, which is the best approximation factor that can be achieved as proved in [7].

First we define a *circle* $C(i,r)$ with radius r around node v_i which consists of all nodes v_j such that $x_{ij} \leq r$. The *cut* of a set of nodes S, denoted by $cut(S)$, is the weight of the positive edges with exactly one endpoint in S:

$$cut(S) = \sum_{|v_j,v_k\cap S|=1,(j,k)\in E^{\langle+\rangle}} c_{jk}. \tag{18}$$

The *cut* of a circle is the cut induced by the set of vertices included in the circle. The *volume* of a set of nodes S, denoted by $vol(S)$, is the weighted distance of the edges with both endpoints in S:

$$vol(S) = \sum_{v_j,v_k\subset S,(j,k)\in E^{\langle+\rangle}} c_{jk}x_{jk}. \tag{19}$$

Therefore, the *volume* of a circle is the volume of $C(i,r)$ including the fractional weighted distance of positive edges leaving $C(i,r)$. In other words, if $(j,k) \in E^{\langle+\rangle}$ is a cut positive edge of circle $C(i,r)$ with $v_j \in C(i,r)$ and $v_k \notin C(i,r)$, then (j,k) contributes $c_{jk} \cdot (r - x_{ij})$ weight to the volume of circle $C(i,r)$. We include an initial volume I to the volume of every circle (i.e., circle $C(i,0)$ has volume I). The correlation clustering algorithm on a weighted proof-correlation graph is shown in Algorithm 4. One advantage of this approach is that the correlation clustering algorithm can handle outliers naturally. Outliers will usually have a cluster of their own. Thus, when the algorithm ends, we can simply discard small isolated clusters.

Algorithm 4. Correlation clustering on proof-correlation graph

```
Input: queue $G$ with all the nodes $v_i \in V$, weight
  function $c_{ij}$ for each edge $(i,j)$;
while $G$ not empty do
    dequeue $v_i \leftarrow G$;
    $r = 0$;
    while $cut(C(i,r)) > c \cdot ln(n+1) \times vol(C(i,r))$ do
        $r = r + min\{(d_{ij} - r) > 0 : v \notin C(i,r)\}$;
    end while
    output $C(i,r)$ as one of the clusters;
    remove $C(i,r)$ and related edges from $G$;
end while
```

6 Performance Evaluations

In this section, we study the feasibility of deploying APPLAUS such as the computation and storage constraint, power consumption, and the proof exchange latency. We also use simulations to evaluate the performance of APPLAUS.

6.1 Prototype Implementation

To study the feasibility of our scheme, we have developed a prototype of APPLAUS based on the techniques presented in the previous sections. The prototype has two software components: client and server. The client is implemented in JAVA on Android Developer Phone 2 (ADP2), which is equipped with 528 MHz chipset, 512 MB ROM, 192 MB RAM, Bluetooth, and GPS module, and running Google Android 1.6 OS. It can communicate with the server anytime through AT&T's 3G wireless data service. The server is implemented on a T4300 2.1 GHz 3 GB RAM laptop. It stores the uploaded location proof records and manages corresponding indices using MySQL 5.0. We use two android phones to communicate with each other to test our solution.

The client code consumes only 80 KB of data memory. When running, less than 2.5 percent of the available

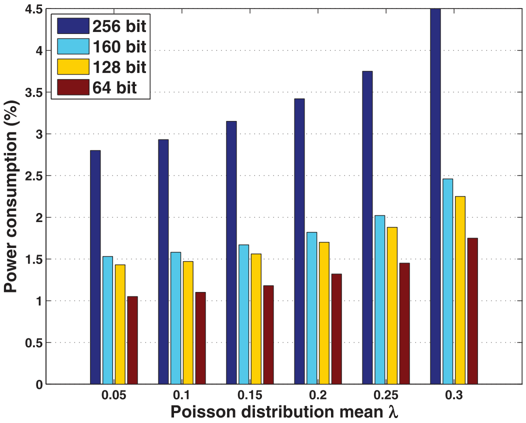


Fig. 6. Power consumptions under different key sizes and Poisson distribution mean λ.

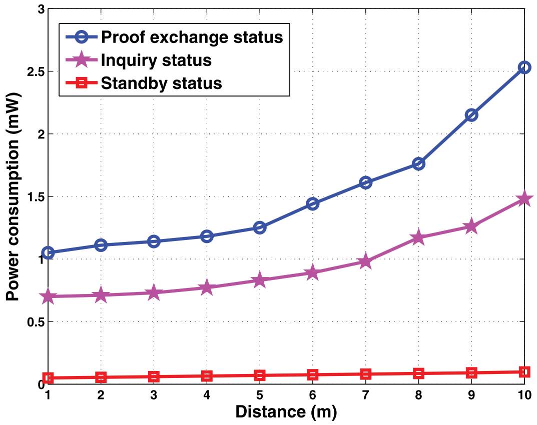


Fig. 7. Power consumption under different Bluetooth status and different communication distance.

memory is used. We measure the CPU utilization of our client code using a monitoring application, which allows one to monitor the CPU usage of all the processes running on the phone. When the application is in standby, the CPU utilization is about 0.5 percent, indicating that listening to incoming Bluetooth inquiries requires very low computation. The CPU utilization goes to 3 and 5 percent, respectively, when communicating with another device and with the server, due to using different communication interfaces. We observe that the CPU utilization reaches the highest level of 10 percent when a location proof packet is generated, in which heavy computations such as authentication and encryption/decryption are involved.

We also study the appropriate size of the public/private key pair. The key size determines the number of bits of the encrypting key as well as the size of the pseudonym. Larger key size can provide better security, but involves more computations and storage resources, as well as power consumption. We use Elliptic Curve Cryptography (ECC) in our implementation as it provides the same functionality and security features as the widely used RSA cryptosystem, but requires much shorter key length for achieving a similar security level. For example, a 160-bit ECC-key attains about the same level of security as a 1,024-bit RSA key. There is always a tradeoff between performance and security level when choosing the key size. In mobile computing scenario where computation and power resource are limited, we tend to assign more weight to performance. Some of the organizations (e.g., recent ECRYPT II recommendation [28]) recommend 160-bit key for ECC, while others recommend 256-bit. Our experimental results in Fig. 6 show that 256-bit key consumes much more power than that of 160-bit key, under different Poisson distribution parameters. As our goal is to find the smallest key size which can achieve general security requirement for light-weight mobile scenario, 160-bit is used to in our prototype. Other cryptography methods or key sizes can be also applied depending on different emphasis of performance or security.

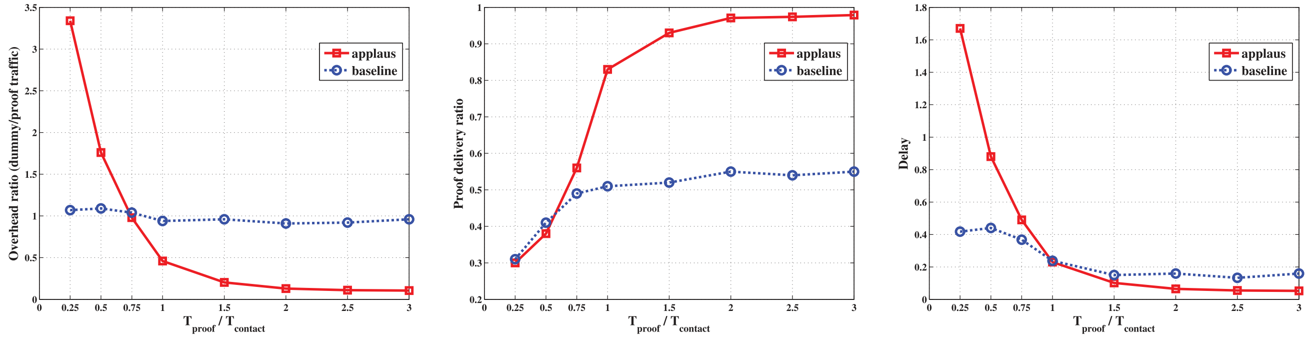
We also show how power consumption affects the daily normal usage of the mobile device. Fig. 6 shows the percentage of power consumption as a function of the Poisson parameter λ used in location proof exchange. As λ becomes larger, the power consumption increases, due to the more frequent location proof updating activities. It is easy to see our scheme will not increase the power consumption too much (with less than 2.5 percent power consumption).

The distance between the phones when they exchange location proofs also affects the latency, where longer distance means longer delay due to the weak signal strength. As has been established in earlier studies [14], more than 80 percent of contact durations are less than 10 seconds, and thus there is no problem for our proof exchange process to be finished within the contact duration. Fig. 7 measures the power consumption under different Bluetooth status. There are three status: *inquiry*, *standby* and *proof exchange*. The inquiry status is used to discover other Bluetooth devices within communication range, and send out proof requests. The inquiry process continues for a prespecified time, until a prespecified number of units have been discovered or until it is stopped explicitly. Bluetooth devices that only listen to inquiry messages are in standby. In our system, inquiry and standby are mutual exclusive at any time. The device enters the proof exchange status when it exchanges location proofs with others. The most frequent status is standby, which consumes less than 0.1 mW of power with any communication distance. The proof exchange status consumes the most amount of power and deteriorates with increasing communication distance; however, it will not appear until the next location proof updating cycle.

6.2 Simulation Results

In our simulations, 1,000 mobile nodes are deployed in a $3\ \mathrm{km} \times 3\ \mathrm{km}$ area. We use the Levy walk mobility model [24] to generate synthetic node contact events. Previous work has shown that the Levy walk model can describe the mobility patterns of a human being relatively well for a campus-sized area. A contact between two nodes happens when the nodes are within 10 m range of each other, which is the communication range of Bluetooth.

For each simulation run, we generate a Levy walk trace with a certain contact rate and initiate the location proof updating process with various time intervals. We repeat this experiment 100 times, choosing different contact rates for each run. Each node has $M = 10$ pairs of 160-bit public/private keys and a Poisson distribution parameter λ, which is used to determine when to change pseudonyms. λ is



(a) Overhead Ratio

(b) Proof Delivery Ratio

(c) Average Delay

Fig. 8. Performance under different ratio of $\omega = T_{proof}/T_{contact}$.

divided into 10 different numbers: $\lambda_1, \lambda_2, \ldots, \lambda_{10}$, where $\lambda = \lambda_1 + \lambda_2 + \cdots + \lambda_{10}$. We define a parameter δ, which is the standard deviation of two consecutive λ_i and λ_{i+1} (assuming λ_i and λ_{i+1} are in nondecreasing order), to control how to cut λ. Another parameter ϵ is chosen by each node as a threshold to determine whether to accept a location proof exchange request based on the user-centric privacy level.

We compare our APPLAUS scheme with a *baseline* scheme. In the baseline scheme, each node does not change pseudonyms based on Poisson distribution. Instead, it uses a constant rate to upload location proofs. Unlike APPLAUS where two nodes mutually exchange location proofs, the baseline scheme only uploads its own location proof if there is a proof available. A dummy message is uploaded when there is no proof available.

In the comparisons, we use three metrics: *message overhead ratio*, *proof delivery ratio*, and *average delay*. The message overhead ratio is defined as the ratio of dummy proof traffic and real location proof traffic. The proof delivery ratio is the percentage of location proof message that is successfully uploaded to the location proof server. The average delay is the time difference between the time when a location proof update is needed and when the location proof message has reached the location proof server.

Fig. 8 shows the performance comparisons between our scheme and the baseline scheme under different ratio of $\omega = T_{proof}/T_{contact}$, where T_{proof} is the required interval between two location proof updates, and $T_{contact}$ is the mean real node contact interval. Fig. 8a shows that APPLAUS outperforms baseline on overhead ratio when ω is larger than 0.75. When $\omega > 1.5$, the overhead ratio of APPLAUS decreases to as low as 0.2. As shown in Fig. 8b, the proof delivery ratio of APPLAUS significantly outperforms the baseline scheme, and it reaches 93 percent when $\omega > 1.5$. From Fig. 8c, we can see that APPLAUS and baseline have similar average delay when $\omega > 1$, in which the delay is measured as the unit of T_{proof}. When $\omega > 1.5$, the delay of APPLAUS becomes lower than 0.15 of T_{proof}. Thus, when $\omega > 1.5$; i.e., when the location proof update interval is at least 1.5 of the contact interval, the performance of APPLAUS reaches an adequate level. The delay will not change too much after $\omega = 2$. Therefore, an appropriate ratio ω should be chosen between 1.5 and 2.

6.3 Privacy Evaluation

In the previous section, we already showed that our location proof updating system has the property of pseudonym unlinkability and statistically strong source location unobservability. In this section, we evaluate the robustness of APPLAUS such as defending against traffic analysis and statistical test.

With our scheme, we consider what the attacker can do to detect real events. Clearly, we cannot prevent the attacker from using any statistical analysis tool, so what we show below are based on the general techniques that can be used by the attacker. We believe other statistic testing methods will render similar results.

We assume that the attacker has enough resources (e.g., storage and computation ability) to collect and analyze location proof updating time by traffic monitoring or by compromising the location proof server. Then, the attacker will try to infer that two pseudonyms belong to the same ID if the distributions of their location proof updating time intervals are identical. To achieve this, the attacker can first conduct some goodness of fit tests such as the Kolmogorov-Simirnov (K-S) test [25] to check whether the probabilistic distributions of the location proof updating time intervals from every pseudonym follow the same Poisson distribution. The attacker then performs the mean test such as Sequential Probability Ratio Test (SPRT) [29] if the distributions can pass the goodness of fit test. Those distributions whose sample means deviate from the true mean beyond a certain degree will be marked as potential abnormal pseudonyms.

To detect if two pseudonyms belong to the same source, the attacker can check whether the two probabilistic distributions of location proof updating time intervals from the two pseudonyms are identical. For the attacker, the hypotheses of the test are:

- H_0—the two pseudonyms belong to the same source.
- H_1—the two pseudonyms belong to different source.

When the attacker makes a decision, there are some risks to get wrong decision. The decision is called a detection, if H_0 is accepted when it is actually true. If H_0 is in fact true, accepting H_1 is a false negative. On the other hand, if H_1 is in fact true, accepting H_0 is a false positive. False positive has no negative effect on privacy since taking two different pseudonyms as the same would not help identifying the real source. We focus on false negative which indicates the percentage of cases that has not been detected by the attackers.

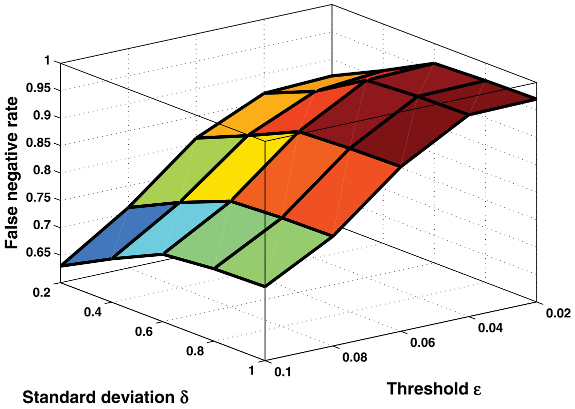


Fig. 9. False negative rate under different parameter of δ and ϵ.

To evaluate false negative, we use the same simulation setup as previous section and repeatedly perform K-S test [22] on the distributions under different standard deviation δ (ranging from 0.2 to 1) and threshold ϵ (ranging from 0.02 to 0.1). Based on the test results, we get the false negative rate as shown in Fig. 9. The x-axis is the threshold ϵ of each user. The smaller ϵ is, the less likely a location proof exchange request will be accepted. The y-axis represents the standard deviation δ of two Poisson distribution means. Obviously, larger δ indicates wider difference between the two Poisson distributions, which is harder for attackers to differentiate. We can observe from the figure that the false negative rate of the attacker's test is high (above 90 percent), especially when $\epsilon > 0.04$ and $\delta > 0.5$, which indicates that as long as the parameters of ϵ and δ are appropriately selected, the privacy of our system can be preserved.

6.4 Collusion Detection

In this section, we evaluate the performance of collusion detection based on betweenness ranking and correlation clustering approaches. We consider three data sets: besides the above mentioned synthetic trace and the MIT reality trace [8], we also crawl live data from Foursquare, one of the most popular location-based social networking services. The data crawling task is to collect correlated user and location data. We select well-connected users (who have hundreds of friends) as seeds to start the crawling. After aggregating the user and location data, we obtain a data set of 58,659 users and 96,219 locations. Note that each user and location in the data set are associated with an address which can be converted to a geographical location (i.e., latitude and longitude) via Google map service.

The quality of betweenness ranking is measured by the ranking score, where we use Mean Average Precision (*MAP*), to measure the ranking performance of the calculated betweenness. *MAP* is the most frequently used summary measure of a ranked retrieval run. In our scenario, it stands for the mean of the precision score after each betweenness is generated:

$$MAP = \frac{\sum_{b=1}^{B} AveP(b)}{B}, \tag{20}$$

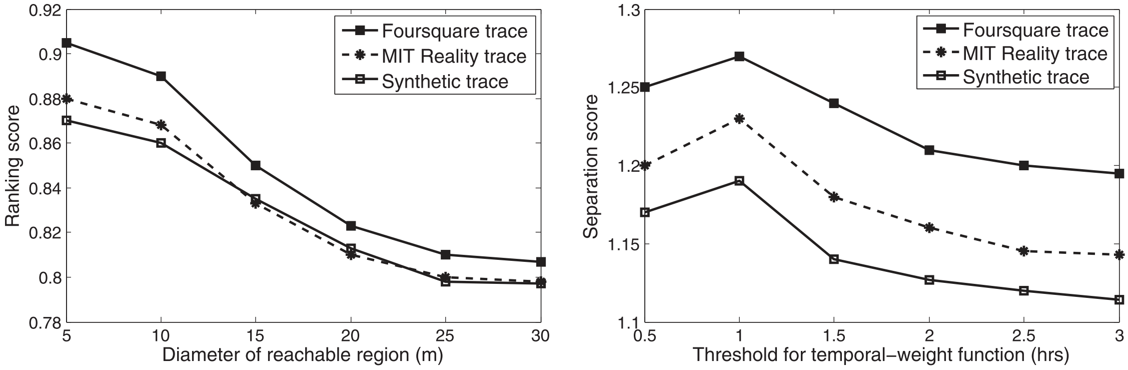
where B is the number of betweenness. Fig. 10a shows the ranking score of the betweenness ranking method.

The quality of correlation clustering is measured by the separation score, which is calculated as a weighted average distance between cluster centers:

$$S_{ep} = \frac{1}{\sum_{i\neq j} N_i N_j} \sum_{i\neq j} N_i N_j D(C_i, C_j), \tag{21}$$

where C_i and C_j are the centers of the ith and jth clusters, N_i and N_j are the number of vetices in the ith and jth clusters, and D is the distance function. Thus, S_{ep} reflects the overall distance between clusters. Higher S_{ep} means better results. Fig. 10b shows the separation score of our temporal-weighted correlation clustering results. Worth to mention, when the threshold of the time delay between location proofs is around 1 hour, our solution achieves the best clustering quality.

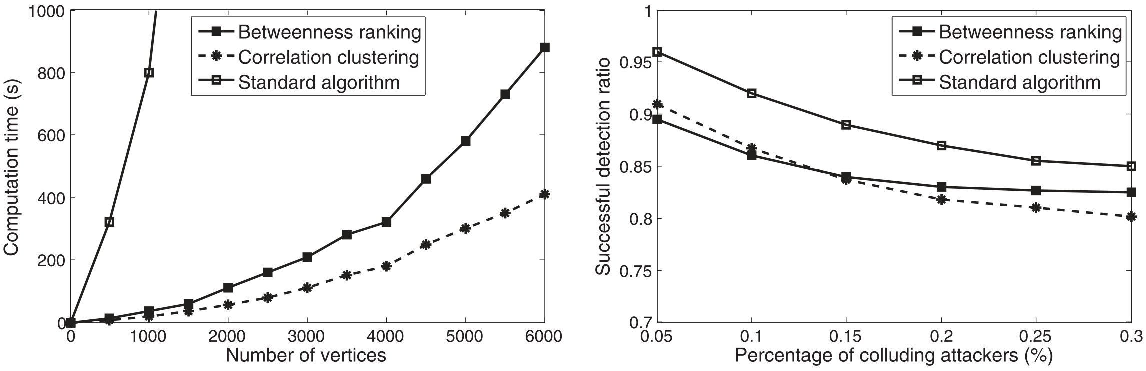
In order to evaluate the performance of these two approaches on detecting colluding nodes, we use computation time and successful detection ratio to measure the efficiency and effectiveness, respectively. We also compare these two approaches with a *standard* algorithm, in which every location proof or pseudonym is verified individually. All approaches are evaluated based on the Foursquare data set. As shown in Fig. 11, the correlation clustering approach runs faster than the betweenness ranking approach, and both are much faster than the standard algorithm. Even though the detection ratio of the clustering approach and the ranking approach is a little bit lower than the standard algorithm, the big difference in computation time is worth the cost.



(a) Ranking score of betweenness ranking

(b) Separation score of correlation clustering

Fig. 10. Metrics of quality evaluation for two approaches.



(a) Computation time with number of vertices

(b) Successful detection ratio

Fig. 11. Performance of colluding detection for two approaches.

7 Related Work

Recently, several systems have been proposed to provide end users the ability to prove that they were in a particular place at a particular time. The solution in [3] relies on the fact that nothing is faster than the speed of light in order to compute an upper bound of a user's distance. Capkun and Hubax [5] propose challenge-response schemes, which use multiple receivers to accurately estimate a wireless node location using RF propagation characteristics. In [15], the authors describe a secure localization service that can be used to generate unforgeable geotags for mobile content such as photos and video. However dedicated measuring hardware or high-cost trusted computing module are required. Saroiu and Wolman [26] propose a solution suitable for third-party attestation, but it relies on a PKI and the wide deployment of WiFi infrastructure. Different from these solutions, APPLAUS uses a peer-to-peer approach and does not require any change to the existing infrastructure. SmokeScreen [6] introduces a presence sharing mobile social service between colocated users which relies on centralized, trusted brokers to coordinate anonymous communication between strangers. SMILE [20], [21] allows users to establish missed connections and utilizes similar wireless techniques to prove if a physical encounter occurred. However, this service does not reveal the actual location information to the service provider thus can only provide location proofs between two users who have actually encountered. APPLAUS can provide location proofs to third-party by uploading real encounter location to the untrusted server while maintaining location privacy.

There are lots of existing works on location privacy in wireless networks. In [11], the authors propose to reduce the accuracy of location information along spatial and/or temporal dimensions. This basic concept has been improved by a series of works [10], [13]. All the above techniques cloak a node's locations with its current neighbors by trusted central servers which is vulnerable to DoS attacks or to be compromised. Different from them, our approach does not require the location proof server to be trustworthy. Xu and Cai [30] propose a feeling-based model which allows a user to express his privacy requirement. One important concern here is that the spatial and temporal correlation between successive locations of mobile nodes must be carefully eliminated to prevent external parties from compromising their location privacy. The techniques in [1], [9] achieve location privacy by changing pseudonyms in regions called mix zones. In this paper, pseudonyms of each node are changed by the node itself periodically following a Poisson distribution, rather than being exchanged between two untrusted nodes. Identifying a fundamental tradeoff between performance and privacy, Shao et al. [27], [31] propose a notion of statistically strong source anonymity in wireless sensor networks for the first time, while Li and Ren [18] and Zhang et al. [32] tried to provide source location privacy against traffic analysis attacks through dynamic routing or anonymous authentication. Our scheme uses similar source location unobservability concept in which the real location proof message is scheduled through statistical algorithms. However, their focus is to generate identical distributions between different nodes to hide the real event source, while our focus is to design distinct distributions between different pseudonyms to protect the real identity.

8 Conclusions

In this paper, we proposed a privacy-preserving location proof updating system called APPLAUS, where colocated Bluetooth enabled mobile devices mutually generate location proofs and upload to the location proof server. We use statistically changed pseudonyms for each device to protect source location privacy from each other, and from the untrusted location proof server. We also develop a user-centric location privacy model in which individual users evaluate their location privacy levels in real time and decide whether and when to accept a location proof exchange request based on their location privacy levels. To the best of our knowledge, this is the first work to address the joint problem of location proof and location privacy. To deal with colluding attacks, we proposed betweenness ranking based and correlation clustering-based approaches for outlier detection. Extensive experimental and simulation results show that APPLAUS can provide real-time location proofs effectively. Moreover, it preserves source location privacy and it is collusion resistant.

References

[1] A.R. Beresford and F. Stajano, "Location Privacy in Pervasive Computing," *IEEE Security and Privacy*, 2003.

[2] U. Brandes, "A Faster Algorithm for Betweenness Centrality," *J. Math. Sociology*, vol. 25, no. 2, pp. 163-177, 2001.

[3] S. Brands and D. Chaum, "Distance-Bounding Protocols," *Proc. Workshop Theory and Application of Cryptographic Techniques on Advances in Cryptology (EUROCRYPT '93),* 1994.

[4] L. Buttyán, T. Holczer, and I. Vajda, "On the Effectiveness of Changing Pseudonyms to Provide Location Privacy in VANETs," *Proc. Fourth European Conf. Security and Privacy in Ad-Hoc and Sensor Networks,* 2007.

[5] S. Capkun and J.-P. Hubaux, "Secure Positioning of Wireless Devices with Application to Sensor Networks," *Proc. IEEE INFOCOM,* 2005.

[6] L.P. Cox, A. Dalton, and V. Marupadi, "SmokeScreen: Flexible Privacy Controls for Presence-Sharing," *Proc. ACM MobiSys,* 2007.

[7] E.D. Demaine, D. Emanuel, A. Fiat, and N. Immorlica, "Correlation Clustering in General Weighted Graphs," *Theoretical Computer Science,* vol. 361, nos. 2/3, pp. 172-187, 2006.

[8] N. Eagle and A. Pentland, "CRAWDAD Data Set mit/reality (v. 2005-07-01)," http://crawdad.cs.dartmouth.edu/mit/reality, July 2005.

[9] J. Freudiger, M.H. Manshaei, J.P. Hubaux, and D.C. Parkes, "On Non-Cooperative Location Privacy: A Game-Theoretic Analysis," *Proc. 16th ACM Conf. Computer and Comm. Security (CCS),* 2009.

[10] B. Gedik and L. Liu, "A Customizable K-Anonymity Model for Protecting Location Privacy," *Proc. IEEE Int'l Conf. Distributed Computing Systems (ICDCS),* 2005.

[11] M. Gruteser and D. Grunwald, "Anonymous Usage of Location-Based Services through Spatial and Temporal Cloaking," *Proc. ACM MobiSys,* 2003.

[12] B. Hoh, M. Gruteser, R. Herring, J. Ban, D. Work, J.C. Herrera, A.M. Bayen, M. Annavaram, and Q. Jacobson, "Virtual Trip Lines for Distributed Privacy-Preserving Traffic Monitoring," *Proc. ACM MobiSys,* 2008.

[13] T. Jiang, H.J. Wang, and Y.-C. Hu, "Location Privacy in Wireless Networks," *Proc. ACM MobiSys,* 2007.

[14] V. Kostakos, "Experiences with Urban Deployment of Bluetooth," presentation given at the Univ. of California, San Diego, Mar. 2007.

[15] V. Lenders, E. Koukoumidis, P. Zhang, and M. Martonosi, "Location-Based Trust for Mobile User-Generated Content: Applications Challenges and Implementations," *Proc. Ninth Workshop Mobile Computing Systems and Applications,* 2008.

[16] M. Li, R. Poovendran, K. Sampigethaya, and L. Huang, "Caravan: Providing Location Privacy for VANET," *Proc. Embedded Security in Cars (ESCAR) Workshop,* 2005.

[17] M. Li, K. Sampigethaya, L. Huang, and R. Poovendran, "Swing & Swap: User-Centric Approaches Towards Maximizing Location Privacy," *Proc. Fifth ACM Workshop Privacy in Electronic Soc.,* 2006.

[18] Y. Li and J. Ren, "Source-Location Privacy Through Dynamic Routing in Wireless Sensor Networks," *Proc. IEEE INFOCOM,* 2010.

[19] W. Luo and U. Hengartner, "Proving Your Location Without Giving Up Your Privacy," *Proc. ACM 11th Workshop Mobile Computing Systems and Applications (HotMobile '10),* 2010.

[20] J. Manweiler, R. Scudellari, Z. Cancio, and L.P. Cox, "We Saw Each Other on the Subway: Secure Anonymous Proximity-Based Missed Connections," *Proc. ACM 10th Workshop Mobile Computing Systems and Applications (HotMobile '09),* 2009.

[21] J. Manweiler, R. Scudellari, and L.P. Cox, "SMILE: Encounter-Based Trust for Mobile Social Services," *Proc. ACM Conf. Computer and Comm. Security (CCS),* 2009.

[22] F.J. Massey Jr., "The Kolmogorov-Smirnov Test for Goodness of Fit," *J. Am. Statistical Assoc.,* vol. 46, no. 253, pp. 68-78, 1951.

[23] A. Pfitzmann and M. Hansen, "Anonymity, Unlinkability, Undetectability, Unobservability, Pseudonymity, and Identity Management—A Consolidated Proposal for Terminology" (Version v0.31 Feb. 15, 2008), http://dud.inf.tu-dresden.de/literatur, 2008.

[24] I. Rhee, M. Shin, K. Lee, and S. Chong, "On the Levy-Walk Nature of Human Mobility," *Proc. IEEE INFOCOM,* 2007.

[25] J.L. Romeu, "Kolmogorov-Simirnov: A Goodness of Fit Test for Small Samples," *START: Selected Topics in Assurance Related Technologies,* 2003.

[26] S. Saroiu and A. Wolman, "Enabling New Mobile Applications with Location Proofs," *Proc. ACM 10th Workshop Mobile Computing Systems and Applications (HotMobile '09),* 2009.

[27] M. Shao, Y. Yang, S. Zhu, and G. Cao, "Towards Statistically Strong Source Anonymity for Sensor Networks," *Proc. IEEE INFOCOM,* 2008.

[28] N. Smart et al., "Ecrypt II Yearly Report on Algorithms and Key Lengths (2009-2010)," ICT-2007-216676, Revision 1.0, 2010.

[29] A. Wald, *Sequential Analysis.* Dover, 2004.

[30] T. Xu and Y. Cai, "Feeling-Based Location Privacy Protection for Location-Based Services," *Proc. 16th ACM Conf. Computer Comm. Security (CCS),* 2009.

[31] Y. Yang, M. Shao, S. Zhu, B. Urgaonkar, and G. Cao, "Towards Event Source Unobservability with Minimum Network Traffic in Sensor Networks," *Proc. First ACM Conf. Wireless Network Security (WiSec),* 2008.

[32] Y. Zhang, W. Liu, and W. Lou, "Anonymous Communications in Mobile Ad Hoc Networks," *Proc. IEEE INFOCOM,* 2005.

[33] Z. Zhu and G. Cao, "APPLAUS: A Privacy-Preserving Location Proof Updating System for Location-Based Services," *Proc. IEEE INFOCOM,* 2011.

[34] Z. Zhu, G. Cao, S. Zhu, S. Ranjan, and A. Nucci, "A Social Network Based Patching Scheme for Worm Containment in Cellular Networks," *Proc. IEEE INFOCOM,* 2009.

Zhichao Zhu received the BS degree from the University of Science and Technology of China in computer science in 2006. He is working toward the PhD degree in the Department of Computer Science and Engineering, Pennsylvania State University. His research interests include the areas of security and privacy in wireless mobile networks, data measurement, social computing, and location sensitive services. He is a student member of the IEEE.

Guohong Cao received the BS degree from Xian Jiaotong University, China, and the MS and PhD degrees in computer science from the Ohio State University in 1997 and 1999, respectively. Since then, he has been with the Department of Computer Science and Engineering at the Pennsylvania State University, where he is currently a professor. His research interests are in wireless networks and mobile computing. He has published more than 150 papers in the areas of wireless network security, wireless sensor networks, vehicular ad hoc networks, cache management, data access and dissemination, and distributed fault tolerant computing. He has served on the editorial boards of the *IEEE Transactions on Mobile Computing*, the *IEEE Transactions on Wireless Communications*, and the *IEEE Transactions on Vehicular Technology*, and has served on the organizing and technical program committees of many conferences. He was a recipient of the US National Science Foundation CAREER award in 2001. He is a fellow of the IEEE.