2010 16th International Conference on Parallel and Distributed Systems

Privacy Protection in Participatory Sensing Applications Requiring Fine-Grained Locations

Kai Dong
Nanjing University
dongkai0918@gmail.com

Tao Gu
University of Southern Denmark
gu@imada.sdu.dk

Xianping Tao*, Jian Lu
Nanjing University
{txp, lj}@nju.edu.cn

***Abstract*—The emerging participatory sensing applications have brought a privacy risk where users expose their location information. Most of the existing solutions preserve location privacy by generalizing a precise user location to a coarse-grained location, and hence they cannot be applied in those applications requiring fine-grained location information. To address this issue, in this paper we propose a novel method to preserve location privacy by anonymizing coarse-grained locations and retaining fine-grained locations using Attribute Based Encryption (ABE). In addition, we do not assume the service provider is an trustworthy entity, making our solution more feasible to practical applications. We present and analyze our security model, and evaluate the performance and scalability of our system.**

***Keywords*-Location privacy, participatory sensing, ABE, k-anonymity.**

I. Introduction

Over the past decade, we have witnessed an explosive growth of mobile devices that are increasingly capable of capturing and transmitting image, sound, location and other data interactively or autonomously. The ubiquity of these devices has brought forth a new class of applications—participatory sensing application [1] (a.k.a. opportunistic sensing applicatioin [2]), e.g., *CarTel* [3], *AnonySense* [4], *Nericell* [5] and *PetrolWatch* [6]. In these applications, mobile phones carried by users collect the information about an urban landscape (e.g., traffic information). The information are then reported to an application server and shared by other users. Since such information include users' spacial and temporal information, the privacy of the users has been put at increased risk. Specifically, the spacial and temporal information in a report may be linked to a particular user, resulting in his privacy being invaded. Thus, the most challenging privacy issue is how to avoid linking between a report and the user.

Simple techniques using pseudonyms or anonymizing reports may not work. For example, if an adversary has a priori knowledge of a user's movement pattern, it is fairly trivial to de-anonymize the reports. A considerable solution is to appoint a trusted anonymizer to guarantee user anonymity. With this anonymizer, the precise location represented by a point in coordinate (i.e., fine-grained location) in each report is generalized to a region in space (i.e., coarse-grained location) where there are at least k users. In this way, it is impossible to distinguish between them. Such technique is known as k-anonymity, and it has been widely adopted in location based services (LBS) [7][8][9][10][11].

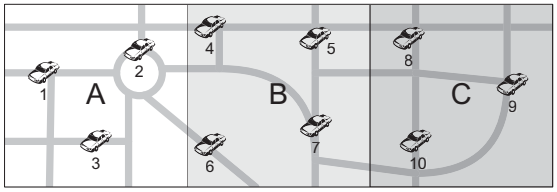

Fig. 1. 3-anonymity: an example

However, the k-anonymity technique only provides coarse-grained location information which may be useless in many participatory sensing applications. For instance, consider a participatory sensing application which reports traffic information in city. As illustrated in Fig. 1, if k-anonymity is used, the intersection along a road, where a reporter is currently located at, will be generalized to a large region (e.g., Region A, B or C). Since the region of a report may cover more than 2 km^2 [4], users cannot ascertain which road is being referred to, leaving this location information little use.

In this paper, we aim to preserve user privacy in participatory sensing applications which require fine-grained locations. An intuitive solution is that a reporter can report coarse-grained location information to the server, and fine-grained location information to the end-users. To achieve this, a reporter may prepare two copies of his location information to be sent to the server—the anonymized, coarse-grained location and the encrypted, fine-grained location. To encrypt fine-grained locations, Public-Key Cryptography (PKC) may do the job. However, in participatory sensing applications, reporters have no prior knowledge of end-users. PKC fails to work in this context because it requires the senders must know the receivers exactly in order to choose the correct public keys.

To overcome the limitation of PKC, in our solution, we adopt the idea of Attribute Based Encryption (ABE) [12]. Using ABE, reporters can encrypt messages based on certain attributes rather than user identities. Since the attributes can be specified by reporters, they can encrypt messages without any prior knowledge of end-users. Such attributes can be, for example, age, gender, occupation and location, etc. Only the users with these attributes can obtain the private keys and decrypt the message. However, attributes in traditional

*Xianping Tao is the corresponding author.

1521-9097/10 $26.00 © 2010 IEEE
DOI 10.1109/ICPADS.2010.127

ABE are static while user locations in participatory sensing applications are changing all the time. Dynamic attributes may cause great difficulties in maintaining their values such as updating a new attribute value and deleting an out-of-date attribute value. We address the key distribution and revocation problems by tolerating masquerading, and demonstrate that our solution is secure against masquerade attacks through our privacy analysis and evaluation.

In summary, this paper makes the following contributions:

- We propose a novel method (P3S) to preserve privacy in participatory sensing applications which require fine-grained locations. P3S is able to preserve location privacy and provide fine-grained location information at the same time.
- We address the key distribution and revocation problems arising from ABE. We introduce a formal privacy model, and demonstrate that even in the worst case, P3S guarantees a certain degree of anonymity for users.
- As a case study, we implement P3S in a road-traffic information service named *VehicleMap*. Base on this implementation, we evaluate the runtime of P3S, and the results show P3S can be applied to other participatory sensing applications in general.

The rest of the paper is organized as follows. We first discuss the related work in Section II. We then describe P3S in Section III, and present the privacy analysis in Section IV. Section V presents our implementation and evaluation results. Finally, we conclude the paper in Section VI.

II. Related Work

Many existing work leverage on the concept of k-anonymity to keep user privacy anonymous. K-anonymity is originally proposed by Sweeney [13][14] in the database community to protect sensitive information from being disclosed. A table satisfies *k*-anonymity if every record is indistinguishable from at least $k-1$ other records with respect to every set of quasi-identifiers. If an aggregation of k reports satisfies k-anonymity, the probability of identifying a user will be theoretically 1/k. Based on k-anonymity, various approaches have been proposed, and they can be broadly classified into the following two categories.

A. Location Blurring

The solutions in this category typically make a precise location blur by generalizing a point in coordinate to a plane in space.

Beresford et al. [15] defined location privacy as the ability to prevent other parties from learning one's current or past location. They introduced a-priori defined Mix Zones to provide anonymity. Users within the same mix zone use pseudonyms to communicate and serve as an anonymity set. The main problem with this system is that there must be enough users in the mix zone to ensure location privacy.

Gruteser et al. [8] applied k-anonymity to protect location privacy, and proposed spatial and temporal cloaking. In this technique, all the requests (from at least k different users) from an area within a certain period of time are managed together as an anonymity set to achieve k-anonymity. However, the area can be large if the user density is low.

The concept of tessellation was first introduced in AnonySense [4][16] to protect user privacy when reporting context information. Tessellation partitions a geographical area into a number of tiles large enough to preserve the users' privacy and each user's location is generalized to a plane in space (i.e., a tile) which covers at least *k* potential users.

In the above approaches, there exists a tradeoff between functionality and privacy, i.e., to ensure privacy protection, they have to sacrifice the granularity of location information. In this work, our goal is to preserve location privacy and provide fine-grained location information at the same time.

B. Fine-grained Location

Little work has been done to achieve privacy protection while providing fine-grained location information. Huang et al. [6] proposed a simple modification to tessellation based on micro-aggregation [17]. They presented an application—PetrolWatch which allows users to automatically collect, contribute and share petrol price information using camera phones. However, in their method, service providers are assumed to be trustworthy, which may not be always true in reality and it is also contradicted with the original intention of tessellation. If a service provider is trustworthy, it is able to provide adequate protection on users' privacy. In this case, no extra protection is needed. In this paper, we alleviate this assumption by assuming service providers to be untrustworthy.

Meyerowitz et al. [10] proposed a solution named CacheCloak to cache service results. Instead of obscuring a user's path by hiding parts of it, they obscure a user's location by surrounding it with other users' paths. When a user requests location data, the CacheCloak server either returns cached data or obtains new data from the location-based service. The CacheCloak always have accurate data available for the user from its cache or from a new anonymous request to the service. However, CacheCloak only makes sense in some location-based services which have relatively static service results, and users should trust the system for securing privacy. For those applications which provide real-time services, this method may have a limitation since the lifetime of the cached data will be very short. Thus, it cannot be applied in participatory sensing applications since the sensing data is changing all the time.

III. P3S: Privacy Protection Based on ABE

In this section, we first give the background of ABE, then illustrate how we use ABE in P3S to achieve privacy protection.

A. Attribute Based Encryption

In ABE, a user's private key is associated with an arbitrary number of attributes expressed as strings. When a party uses ABE to encrypt a message, they specify an associated access structure—a logical expression over attributes. For instance,

{"student of Nanjing university" AND "major: computer science"} is an access structure. A user will only be able to decrypt a cipher text if that user's attributes pass through the cipher text's access structure. Suppose Alice chooses to encrypt a message with access structure {"student of Nanjing university" AND "major: computer science"}, only the students of Nanjing university whose major is computer science can decrypt this message.

There are four kinds of keys involved in ABE: 1) a master public key which is published by PKG; 2) a master private key which is retained by PKG; 3) public keys computed by the reporters; 4) corresponding private keys generated by the PKG. ABE works as follows. First, PKG publishes a master public key, and retains the corresponding master private key. Given the master public key, any party can compute a public key corresponding to one or more attributes by combining the master public key with the attribute values. PKG uses the master private key to generate the private key for users with certain attributes. To obtain a corresponding private key, the users with these attributes contact PKG and use the attribute values as authorization.

In ABE, attributes are static. However, attributes in P3S are dynamic. For example, the locations of reporters are changing over the time. Such dynamic attributes may create more overhead in maintaining their values such as updating a new attribute value and deleting an out-of-date attribute value. This limitation makes key distribution difficult and creates revocation problems. We will address these limitations in Section III-E.

B. Defining Access Structures

In P3S, each end-user owns his attributes, and each reporter has his access structure. Before a reporter uses ABE to encrypt a report, he should first define the access structure to the report. P3S provides two ways for a reporter to define an access structure.

Defined by System: The system should provide a default access structure. In P3S, a user is authorized by the system to access fine-grained location information based on trust management. In this case, security policies and security credentials are needed [18] to indicate which kinds of actions are secure and what kinds of users are trusted.

Defined by Reporter: Alternatively, reporters are able to define their own access structures by choosing an arbitrary number of attributes indexed by PKG. The chosen attributes are different in different applications. We will introduce what attributes we choose in Section V-B.

C. P3S Overview

P3S consists of the following entities, as illustrated in Fig. 2.

- Data Collection Entity: It collects sensing data from reporters and protects their location privacy using k-anonymity.
- Data Sharing Entity: It provides privacy protection for end-users whose location information are needed to access location-based services.

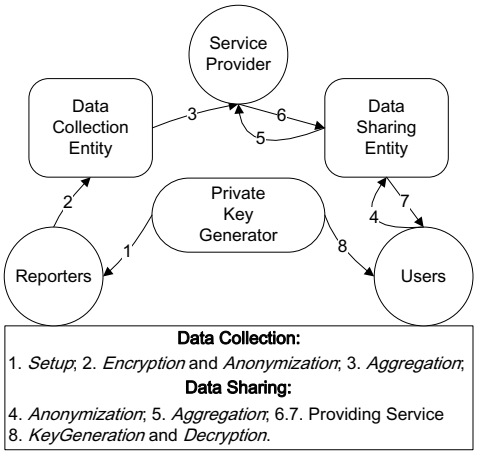

Fig. 2. P3S method

- Private Key Generator (PKG): This is the only trusted entity in P3S which provides public keys to reporters and private keys to end-users. A trusted third party is necessary, and a similar assumption can be found in many other work [15][7][10].

Data collection and sharing entities might not be trusted by users, thus they only get coarse-grained location information.

D. P3S Method

P3S adopts the basic idea of ABE. It consists of the following algorithms. To simplify our illustration, we name users transmitting reports "reporters", and users accessing services "end-users".

Setup: This is a randomized algorithm enforced by PKG. It does not take any input other than the implicit security parameter. It outputs a master public key $\mathcal{PK}$ and a master private key $\mathcal{MK}$.

$$Setup() \Longrightarrow \mathcal{PK} + \mathcal{MK}$$

Encryption: This is a randomized algorithm enforced by reporters. The inputs are as follows: a fine-grained location $\mathcal{FL}$ in plain text, a set of attributes $\alpha\mathcal{S}$ which is defined by the system or the reporters, and the public parameters $\mathcal{PK}$. It outputs the cipher text of the fine-grained location $\mathcal{EL}$.

$$Encryption(\mathcal{FL}, \alpha\mathcal{S}, \mathcal{PK}) \Longrightarrow \mathcal{EL}$$

Anonymization: This algorithm is enforced by reporters. The inputs are as follows: a fine-grained location $\mathcal{FL}$ in plain text and a parameter $\mathcal{K}$. It outputs the coarse-grained location $\mathcal{AL}$.

$$Anonymization(\mathcal{FL}, \mathcal{K}) \Longrightarrow \mathcal{AL}$$

Aggregation: This is a randomized algorithm enforced by the data collection entity. The inputs are $(\mathcal{EL}, \mathcal{AL})$ pairs, and a parameter $\mathcal{L}$. It outputs a series of reports $\mathcal{RS}$.

$$\mathcal{LP} \xrightarrow{def} (\mathcal{EL}, \mathcal{AL})$$

$$Aggregation(\mathcal{LP}_1, \mathcal{LP}_2, \mathcal{LP}_3, ...\mathcal{L}) \Longrightarrow \mathcal{RS}$$

KeyGeneration: This is a randomized algorithm enforced by PKG. The inputs are as follows: a set of attributes $\mathcal{AS}$, a master private key $\mathcal{MK}$ and a master public key $\mathcal{PK}$. It outputs a decryption key $\mathcal{DK}$.

$$KeyGeneration(\mathcal{AS}, \mathcal{MK}, \mathcal{PK}) \Longrightarrow \mathcal{DK}$$

Decryption: The inputs of this algorithm are as follows: the cipher text of location $\mathcal{EL}$ that was encrypted under $\alpha\mathcal{S}$, a decryption key $\mathcal{DK}$ for $\mathcal{AS}$ and a master public key $\mathcal{PK}$. It outputs a fine-grained location $\mathcal{FL}$ in plain text if $\alpha\mathcal{S} \subseteq \mathcal{AS}$.

$$Decryption(\mathcal{EL}, \mathcal{DK}, \alpha\mathcal{S}, \mathcal{AS}) \xrightarrow{\alpha\mathcal{S}\subseteq\mathcal{AS}} \mathcal{FL}$$

These algorithms work in the following way as shown in Fig. 2. First, PKG enforces *Setup* to publish a master public key $\mathcal{PK}$, and retains the corresponding master private key $\mathcal{MK}$. Then, a reporter downloads $\mathcal{PK}$ from PKG, and uses *Encryption* and *Anonymization* to prepare the two copies of his location. The data collection entity enforces *Aggregation* to aggregate reports and send them to the service provider. The service provider can use the coarse-grained location for location-based services, but it is not able to decrypt the cipher text of the fine-grained location. Only the users satisfying $\alpha\mathcal{S} \subseteq \mathcal{AS}$ are able to get the fine-grained location $\mathcal{FL}$ through *Decryption*.

E. Key Management

In a traditional ABE system, a reporter defines his access structure and assigns the attribute values for the receivers. In this case, the system does not care whether a receiver's attribute value is real or not. However, in most participatory sensing applications, the location of each user changes constantly due to mobility, so does his attribute. In this case, it is impossible for an individual reporter to assign the dynamic attribute values to potential users. Thus PKG should authenticate and testify the attributes of all the users. Maintaining users' attribute values such as updating a new attribute value and deleting an out-of-date attribute value is thus a challenging task. This is also known as the key distribution and revocation problems.

To address the above limitation of ABE in a scenario which involves dynamic attributes, instead of requiring PKG to authenticate and testify the dynamic location attribute of each users, we make a concession to tolerate users' masquerading their locations. However, limiting each user can obtain only one private key in each time interval. In a time interval t_i, a user reports an arbitrary value v_i of his attribute to PKG. PKG generates a private key based on the union of $v_i|t_i$, and passes the key back to this user. In this case, the key distribution problem is solved since PKG does not need to worry about the attribute values; and the key revocation problem is solved since each private key can only decrypt the information generated in a time interval t_i.

We summarize our key management using the following lemma.

Lemma 3.1: One can not be in two places at once.

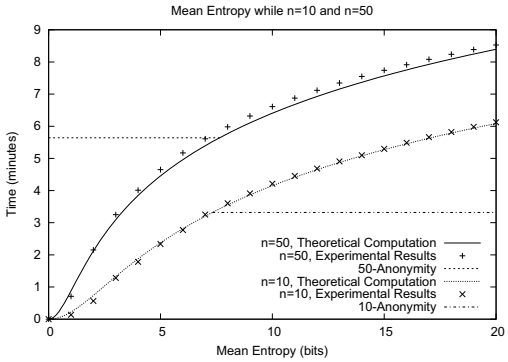

Fig. 3. Mean entropy at different user density

The key management in P3S can not guarantee the authenticity of each user's attribute values. This limitation may rise privacy concerns under attacks, especially masquerade attacks. In the following section, we demonstrate how P3S secures privacy under masquerade and collusion attacks.

IV. Privacy Analysis

In this section, we first define our privacy metric, and then introduce an attacker model and show P3S is secure under different attacks.

A. Privacy Metrics and Evaluation

In location sensitive applications, location privacy is typically measured by location entropy [19]. Location entropy gives a precise quantitative measure of an attacker's uncertainty. It is defined as the number of bits E,

$$E = -\sum P(x, y) \cdot \log_2 P(x, y) \tag{1}$$

for the probability $P(x, y)$ that a user is at location (x, y). By Equation 1, 2^E locations with an equal likelihood will result in E bits of entropy, thus for k-anonymity, we have $k = 2^E$. The higher E is, the more uncertain a hostile observer will be about the true answer, and therefore the higher anonymity we achieve. Similar privacy measurements can be found in [15][10][20].

In participatory sensing applications, location entropy is determined by several factors: the distribution of tiles, the speed of a vehicle, the length of the time interval. For simplicity, we assume that tiles are uniformly distributed to estimate the mean entropy of an urban. Let n denote the number of users in a tile. Let t denote the length of the time interval, V denote the vehicular speed of a user u, and a denote the area of a tile. The location entropy for this user can be calculated as follows:

$$E_u = \log_2[\int_1^{Vt/a^{1/2}} 8x \cdot \bar{n} \cdot dx \cdot \bar{n} + 1] \tag{2}$$

To make further analysis, we set the vehicle speed in an urban to $8km/h$ according to the result in [10]. We set the mean area of a tile to 3.25 km^2 based on the statistic data we

collected from 28 cities in China (more details will be given in Section V). We run the experiment as follows. We first generate an urban area of Nanjing with 18 km x 18 km from GoogleMap. Initially, vehicles are placed randomly in this area, and start to move to random destinations. We compute the mean entropy, and the result is shown in Fig. 3.

From Fig. 3, we observe that when the time interval is short ($< 7min$), the mean privacy entropy in P3S is lower than the theoretical entropy ($log_2 n$) in n-anonymity. It means that the probability of distinguishing a user is more than $1/n$. This is because a user can not move far in a short period of time. In this case, among all the n candidate locations, the nearest one is most likely his location.

B. Attacker Model

From the above analysis, we conclude that the location entropy may reduce in the presence of an attack. Before we analyze possible attacks in P3S, we define our attacker models as follows.

1) Link Attack: Given a set $\mathcal{L}$ of locations *l*, we say that $g : \mathcal{L} \Longrightarrow \mathcal{L}$ is a generalization function. Let *l* denote an accurate location, and *l'=g(l)* denote the generalization of *l* that is forwarded to SP.

Definition 4.1: Let $\mathcal{U}$ denote the set of users *u*. A link attack based on knowledge Γ is a function $Lin_\Gamma : \mathcal{L} \times \mathcal{U} \Longrightarrow [0, 1]$, such that, for each generalized location *l'*, we have:

$$\sum_{u \in \mathcal{U}} Lin_\Gamma(l', u) = 1 \tag{3}$$

By Definition 4.1, link attacks can be specified in which, given a location *l'*, the candidate users have different probabilities to locate at *l'*. We illustrate it using the following example.

Example 4.1: Suppose Alice is parking near a fashion shop, and Bob is driving to a palaestra. If they happened to be in the same tile, they will have different probabilities to locate at each of the locations due to their difference in gender. Based on common senses such as women love shopping and men enjoy in football games, if Alice and Bob are the only two users in this tile, it is fair easy to deduce that Alice is near the shop and Bob is near the palaestra.

2) Track Attack: Given a set $\mathcal{U}$ of users *u*, we say that $p : \mathcal{U} \Longrightarrow \mathcal{U}$ is a pseudonym function. Let *u* denote a real user, and *u'=p(u)* denote the pseudonym of *u* that is forwarded to SP.

Definition 4.2: A track attack based on knowledge Γ is a function $Tra_\Gamma : \mathcal{L} \times \mathcal{U} \Longrightarrow [0, 1]$, such that for each anonymized user *u'*, we have:

$$\sum_{l \in \mathcal{L}} Tra_\Gamma(l, u') = 1 \tag{4}$$

By Definition 4.2, track attacks can be specified in which, given an pseudonymized user *r'*, he has different probabilities to locate at the candidate locations. We illustrate it using the following example.

Example 4.2: Suppose $user1$ is parking near a fashion shop, and $user2$ is driving to a palaestra. Here $user1$ and $user2$ are pseudonyms to prevent the link attacks which arise from gender differences. However, a few minutes ago, $user1$ was at Alice's personal garage near the shop, and $user2$ was on Bob's driveway near the palaestra. In this case, if Alice and Bob are the only two users in this tile, attackers can deduce with high confidence that Alice is $user1$ and Bob is $user2$.

3) Attacker Model: Combining the above two attacks, we define our attacker model as follows.

Definition 4.3: $Att_\Gamma : \mathcal{L} \times \mathcal{U} \Longrightarrow [0, 1]$, such that:

$$\sum_{l \in \mathcal{L}} Att_\Gamma(l, u') \times \sum_{u \in \mathcal{U}} Att_\Gamma(l', u) = 1 \tag{5}$$

Example 4.3: In a participatory sensing application, suppose that we have two users—Alice and Bob. They use $user1$ and $user2$ as their pseudonyms, and report their locations every a few minutes. The locations they report can be expressed as l'_1, l'_2, ..., l'_n, where l' indicates the generalized location in their report. If 2-anonymity is used, l'_i can be expressed as $<l_{iA}, l_{iB}>$. We analyze our privacy model in the following three cases.

Case 1 (link attack): Under this attack, an attacker can deduce that Alice is located at l'_1, l'_2, ..., l_{iA}, ..., l'_n, and Bob is located at l'_1, l'_2, ..., l_{iB}, ..., l'_n. The attacker knows the users' location at i but the users' locations still satisfies 2-anonymity at other time intervals.

Case 2 (track attack): Under this attack, an attacker can deduce that A user is located at l'_{1A}, l'_{2A}, ..., l'_{nA}, and another user is located at l'_{1B}, l'_{2B}, ..., l'_{nB}. Although the attacker knows that a sequence of locations l'_{1A}, l'_{2A}, ..., l'_{nA} belongs to a user and l'_{1B}, l'_{2B}, ..., l'_{nB} belongs to another, he does not know which one is Alice or Bob.

Case 3 (combination attack): Under a combination attack (both link attack and track attack), an attacker can deduce that Alice is located at l_{1A}, l_{2A}, ..., l_{nA}, and Bob is located at l_{1B}, l_{2B}, ..., l_{nB}. Users' location privacy is invaded in this case.

An occasional disclosure by a link between a user and an accurate location should not be considered as an attack. For example, suppose an attacker sees Alice near a shop. This kind of disclosure is inevitable, and we should focus on the attacks arising from the system, i.e., we should prevent him from knowing the historical and future location information of Alice. In another word, we need to prevent tracking at any time in any tiles for any users. Base on this understanding, we analyze the performance of P3S under different attacks in the following sections.

C. Masquerade Attack

Our key management in P3S tolerates masquerading, thus an attacker may obtain a private key by reporting a masquerade location to PKG. This key can then be used for decryption. When user density is low, or attackers know certain background knowledge, it may be easy to link a user to a certain location. We name this attack the masquerade attack. The masquerade attack exists due to the limitation of handling dynamic attributes in ABE.

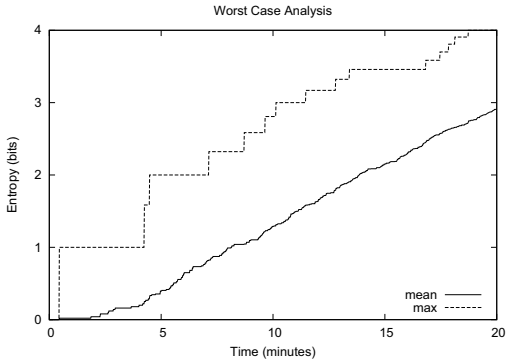

Fig. 4. Lowest entropy: $n = 1$

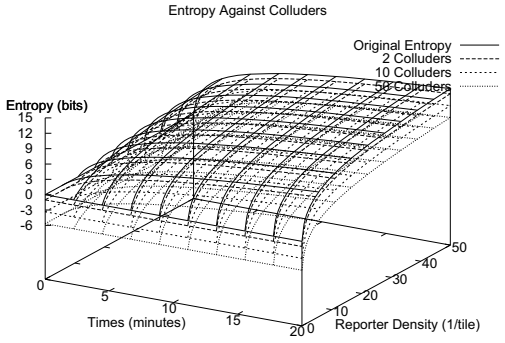

Fig. 5. Collusion attacks: entropy against different number of colluders

P3S prevents the masquerade attack by preventing tracking users. Although one can masquerade to get the fine-grained location information, he is not able to know which reporter is located at this location because he can not track the reporter. In P3S, a user can obtain only one private key at a time (according to Lemma 3.1). If an attacker uses fake attribute values, it will be difficult to obtain the correct private key for a specific user. Taking k-anonymity as an example, the probability of identifying a specific user is $1/k$. If each attribute has m different values on average, the probability of identifying a specific user is $1/m$ in the case that only one attribute is involved, and the probability decreases to $1/m^n$ in the case that an union of n attributes is involved.

In P3S, suppose there are very few users (e.g., less than 10 in a tile), we analyze masquerade attacks in the worst case where each user in a tile can be distinguished and linked to a precise location (e.g., $n = 1$). Fig. 4 shows that, in the worst case, P3S still provides a certain degree of anonymity even if there is only 1 user in each tile. Suppose a user in Tile L_0 move to one of the 10 possible tiles in the next time interval. In this case, the probability of tracking this user is $1/10$. Suppose an attacker masquerade his location as L_i, he will lost his target if the user moves to L_j ($i \neq j$).

D. Collusion Attack

In a collusion attack, multiple attackers may collude with each other to obtain private keys to decrypt fine-grained locations in a large region which may cover many tiles. We conduct an experiment to estimate the mean location entropy under a collusion attack. Each attacker masquerades a different location surrounding a normal user. In this case, to track a user, attackers are able to get as much information surrounding him as possible. Fig. 5 illustrates that, under collusion attacks, the mean entropy is reduced rapidly. When there are 50 colluders, the entropy is reduced by 5 to 6 bits.

We prevent collision attacks using the following idea. When the redundance rate $> r$ (i.e., many redundant reports in bustling places), we can lengthen the sensing interval for reporters in related tiles to increase the location entropy. When the redundance rate $< r$, if there are more than s users querying for different tiles not far from each other, some of these queries will be denied to guarantee a lowest bound of entropy e for all users. It can be calculated in the equation as follows.

$$e_u = \log_2[\int_1^{Vt/a^{1/2}} 8x \cdot \bar{n} \cdot dx \cdot \bar{n}/s + 1] \qquad (6)$$

V. Implementation and Runtime Performance

We implement P3S in a participatory sensing application (*VehicleMap*) which provides road-traffic information. In this section, we first present our implementation, then evaluate the performance of P3S.

A. Implementation in VehicleMap

VehicleMap provides mobile users with real-time traffic information which are reported by different vehicles. A report containing the vehicle's acceleration, speed and location is generated by a mobile phone and sent to the service provider. The service provider can then interpret the data, conclude a jam and notify the users querying this service.

Fig. 6 shows a screen-shot of the city of Nanjing in China (taken from Google Maps at 14:58 on Nov 23, 2009). The four colors—black, brown, yellow and green—are used to indicate the live traffic from the slowest to the fastest. Suppose a patient is currently at location A (a green thumb-tack in $T2$), and wants to find a path with least traffic jams to a hospital located at location B in $T5$. The blue line shows the shortest path from A to B, however this path was indicated with the slowest traffic (the slow traffic areas are marked with red circles 1 to 4).

Using traditional k-anonymity techniques, the fine-grained location of a jam is blurred to a tile and users only obtain the traffic information as "there are jams in $T3$, $T7$, $T11$ and $T15$", thus this patient will have to bypass all these tiles. If there is also a jam at P, the previous techniques fail to find a path which can bypass all the jams. If P3S is employed, $L2$ will be recommended as the desired path with no jams.

B. Defining Access Structure

We use location and time for building up access structure. The location information of a jam in a user's neighborhood

Fig. 6. *VehicleMap* (A: patient, B: hospital)

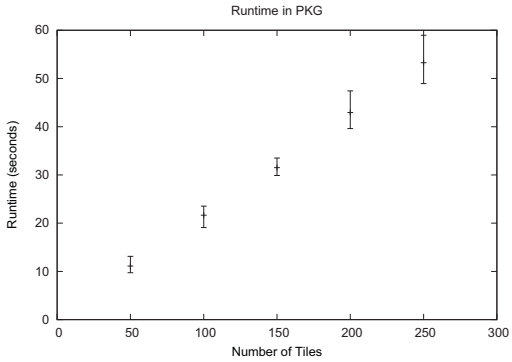

Fig. 7. Runtime of the *KeyGeneration* function in PKG

is necessary to him. We use a need-to-know principle as follows: the users are only allowed to query for the precise information nearby. This principle is designed based on the following observations: Suppose a user requests at location A with the destination location B, 1) Users need to know the traffic information around them, e.g., neighboring streets; 2) When a user is far from his destination, he pays more attentions on the driving direction, instead of which roads or streets he will drive through; 3) Traffic information is quite dynamic, to pre-fetch traffic information of a certain area is often useless; 4) Location privacy is not so sensitive to users nearby.

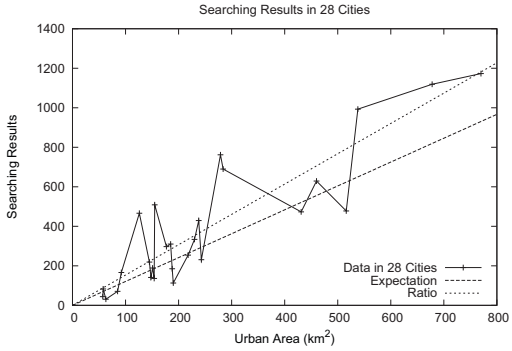

Fig. 8. Searching results and the urban areas of 28 cities in China.

C. Runtime Performance

We focus our evaluation on the runtime and scalability of P3S. In *VehicleMap*, the private keys for the same tile which are generated at different time intervals will differ. The private key which is used to decrypt the fine-grained location of a jam happened at 5pm will not work at another time, e.g., 6pm. This requires extensive computation which may affect the performance of P3S seriously. Moreover, previous studies show that ABE operations are about 100-1000 times slower than those of RSA [21]. Hence, the runtime performance of P3S is critical to real applications.

1) Runtime of PKG: P3S consists of six functions. *Setup* runs off-line. *Aggregation* runs on the data collection and sharing entities, and its runtime is determined by the time of waiting for aggregating reports and queries. *KeyGeneration* runs on PKG, and it is the most time-consuming process as all the private keys are computed in this function. Hence, we focus on evaluating *KeyGeneration*.

We run *KeyGeneration* on a laptop computer with a 1.86 GHz processor and 2 GB of RAM. In the experiment, we generate the attribute values pseudo-randomly, and the average length of the attribute values is 16 bytes. Fig. 7 shows the runtime over 50-250 tiles.

The runtime of *KeyGeneration* is the sum of the time of generating the privacy key for each tile. If more than one user queries for the same tile during the same time interval, PKG only needs to compute the private key once for all these users. In the worst case, PKG has to compute private keys for all the tiles in a city during a time interval.

2) Scalability: We evaluate the scalability of P3S with regards to the number of tiles (i.e., the number of times running *KeyGeneration* in the worst case). To estimate the quantity of the tiles, we use Google Maps and search for "street office" in 28 cities in China. There is typically a street office for several streets, and each street office corresponds to a predefined tile. The number of searching results indicates the quantity of the tiles to some extent.

We compare the urban areas, the searching results and the number of street offices in Nantong, Nanjing and Chengdu. For example, in Nanjing, we obtain 231 searching results and the urban area of Nanjing is about 243 km^2. In fact, there are 46 street offices [22], hence the ratio of the number of street offices to the searching results is $46/231 \doteq 1/5$. Finally, we conclude $1/5$ of the searching results on average as an indication to the real number of street offices.

Fig. 8 shows the number of searching results and the urban areas of 28 cities in China. The black line indicates the distribution of searching results. The dotted line shows the ratio of the total number of searching results to the total number of urban areas. The broken line shows the expectation which is obtained by the average ratio over all the cities. We obtain the mean area of tiles for different cities from 1.35 to 10.16 km^2, and the mean area for all the cities 3.25 km^2. The

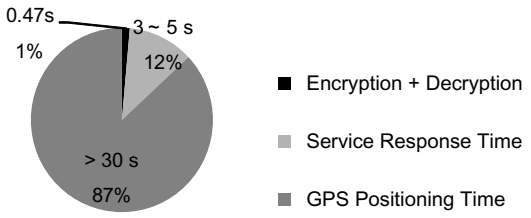

Fig. 9. Runtime performance of P3S on iPhone

number of searching results falls in the range of 31 to 1173 for most of the cities. Thus, we use 1200 as the upper limit of the searching results, and we obtain the upper limit of tiles as $1200 * 1/5 = 240$. Combining with the runtime performance shown in Fig. 7, we observe that, in the worst case, it costs 50 seconds to generate all the private keys in a time interval. We conclude that the time interval should be longer than 1 min in a large city.

3) Runtime Performance on Mobile Devices: To evaluate the feasibility of P3S on mobile devices, we deployed *VehicleMap* on a mobile device, and evaluate the runtime of *Encryption*, *Decryption* and *Anonymization*. We use a first-generation iPhone with a 620MHz ARM processor, and the primary package from Cydia installer such as Classpath, IPhone/Java, JamVM, and Jikes for our implementation.

Fig. 9 shows the runtime of *VehicleMap* in the worst case. The result shows that GPS positioning takes up the most of the runtime (> 30 seconds). The three functions take 0.47 seconds, which is the sum of the time for reporters to generate two copies of a location (the cipher text of fine-grained location and coarse-grained location), and the time for end-users to decrypt the cipher text. The time of waiting for the response from the service provider is about 3 to 5 seconds. The above results show P3S only takes about one percent of the entire time, demonstrating its feasibility for real-life deployments.

VI. Conclusion

In this paper, we propose P3S to preserve location privacy in participatory sensing applications requiring fine-grained location information using ABE. We validate and evaluate P3S using *VehicleMap*. The results show that not only P3S secures user privacy, but also it is feasible in real deployment.

For our future work, we plan to improve the robustness of P3S against a single point of failure. We can use a distributed architecture in which each of the three computation entities (data collection entity, data sharing entity and PKG) is distributed or replicated.

Acknowledgment

This work was supported by the University of Southern Denmark Research Fund, National 973 Program of China under grant 2009CB320702, Natural Science Foundation of China under Grants 60736015 and 61073031, and Pandeng Plan of Jiangsu under grant BK2008017.

References

[1] J. Burke, D. Estrin, M. Hansen, A. Parker, N. Ramanathan, S. Reddy, and M. Srivastava, "Participatory sensing," in *World Sensor Web Workshop*, 2006, pp. 1–5.

[2] A. Campbell, S. Eisenman, N. Lane, E. Miluzzo, and R. Peterson, "People-centric urban sensing," in *Proceedings of the 2nd annual international workshop on Wireless internet*. ACM New York, NY, USA, 2006.

[3] B. Hull, V. Bychkovsky, Y. Zhang, K. Chen, M. Goraczko, A. Miu, E. Shih, H. Balakrishnan, and S. Madden, "CarTel: a distributed mobile sensor computing system," in *Proceedings of the 4th international conference on Embedded networked sensor systems*. ACM, 2006, p. 138.

[4] A. Kapadia, N. Triandopoulos, C. Cornelius, D. Peebles, and D. Kotz, "AnonySense: Opportunistic and privacy-preserving context collection," *Lecture Notes in Computer Science*, vol. 5013, p. 280, 2008.

[5] P. Mohan, V. Padmanabhan, and R. Ramjee, "Nericell: Rich monitoring of road and traffic conditions using mobile smartphones," in *Proceedings of the 6th ACM conference on Embedded network sensor systems*. ACM New York, NY, USA, 2008, pp. 323–336.

[6] K. Huang, S. Kanhere, and W. Hu, "Towards Privacy-Sensitive Participatory Sensing."

[7] C. Bettini, S. Mascetti, X. Wang, and S. Jajodia, "Anonymity in location-based services: towards a general framework," in *Mobile Data Management, 2007 International Conference on*, 2007, pp. 69–76.

[8] M. Gruteser and D. Grunwald, "Anonymous usage of location-based services through spatial and temporal cloaking," in *Proceedings of the 1st international conference on Mobile systems, applications and services*. ACM New York, NY, USA, 2003, pp. 31–42.

[9] H. Kido, Y. Yanagisawa, and T. Satoh, "An anonymous communication technique using dummies for location-based services," in *Pervasive Services, 2005. ICPS'05. Proceedings. International Conference on*, 2005, pp. 88–97.

[10] J. Meyerowitz, R. Choudhury, *et al.*, "Hiding stars with fireworks: location privacy through camouflage," in *Proceedings of the 15th annual international conference on Mobile computing and networking*. ACM, 2009, pp. 345–356.

[11] G. Zhong and U. Hengartner, "A distributed k-anonymity protocol for location privacy," in *Proc. of the Seventh IEEE International Conference on Pervasive Computing and Communication (PerCom 2009), Galveston, TX, USA (March 2009)*.

[12] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proceedings of the 13th ACM conference on Computer and communications security*. ACM, 2006, p. 98.

[13] L. Sweeney, "Achieving k-anonymity privacy protection using generalization and suppression," *International Journal of Uncertainty Fuzziness and Knowledge-Based Systems*, vol. 10, no. 5, pp. 571–588, 2002.

[14] ——, "k-anonymity: A model for protecting privacy," *International Journal Of Uncertainty Fuzziness And Knowledge Based Systems*, vol. 10, no. 5, pp. 557–570, 2002.

[15] A. Beresford and F. Stajano, "Location privacy in pervasive computing," *IEEE Pervasive Computing*, pp. 46–55, 2003.

[16] C. Cornelius, A. Kapadia, D. Kotz, D. Peebles, M. Shin, and N. Triandopoulos, "AnonySense: Privacy-aware people-centric sensing," 2008.

[17] J. Domingo-Ferrer and V. Torra, "Ordinal, continuous and heterogeneous k-anonymity through microaggregation," *Data Mining and Knowledge Discovery*, vol. 11, no. 2, pp. 195–212, 2005.

[18] M. Blaze, J. Feigenbaum, J. Lacy, *et al.*, "Decentralized trust management," in *IEEE Symposium on Security and Privacy*. IEEE COMPUTER SOCIETY, 1996, pp. 164–173.

[19] C. Shannon, "A mathematical theory of communication," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 5, no. 1, p. 55, 2001.

[20] L. Pareschi, D. Riboni, and C. Bettini, "Protecting users anonymity in pervasive computing environments," in *Sixth Annual IEEE International Conference on Pervasive Computing and Communication (PERCOM08), IEEE Computer Society*, 2008, pp. 11–19.

[21] R. Baden, A. Bender, N. Spring, B. Bhattacharjee, and D. Starin, "Persona: an online social network with user-defined privacy," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 4, pp. 135–146, 2009.

[22] http://www.chinayearbook.com, "Nanjing stat. yearbook," 2008.