Privacy-preserving Publishing of Pseudonym-based Trajectory Location Data Set

Ken Mano
NTT Corporation
Email: *mano.ken@lab.ntt.co.jp*

Kazuhiro Minami
Institute of Statistical Mathematics
Email: *kminami@ism.ac.jp*

Hiroshi Maruyama
Institute of Statistical Mathematics
Email: *hm2@ism.ac.jp*

***Abstract*—Anonymization is a common technique for publishing a location data set in a privacy-preserving way. However, such an anonymized data set lacks trajectory information of users, which could be beneficial to many location-based analytic services. In this paper, we present a dynamic pseudonym scheme for constructing alternate possible paths of mobile users to protect their location privacy. We introduce a formal definition of location privacy for pseudonym-based location data sets and develop a polynomial-time verification algorithm for determining whether each user in a given location data set has sufficient number of possible paths to disguise the user's true movements. We also provide the correctness proof of the algorithm.**

***Keywords*-Location privacy; Privacy-preserving data publishing; Pseudonymization;**

I. Introduction

Nowadays a huge number of people are using mobile devices equipped with a GPS receiver, and so it has become feasible to keep track of people's movements over a wide area by collecting GPS data from those mobile devices. Such a large volume of location data gives us a precise global view of people's mobility patterns, and we can thus support analytic location-based services, such as real-time traffic monitoring [1] and urban planning for future sustainable cities [2].

However, due to the significant concern about location privacy [3], the sharing of mobile users' location traces has largely been restricted to anonymized data sets where users' identities are removed. We usually need to follow the practice of ensuring k-anonymity [4], which degrades the granularity of location data to ensure that every location contains more than k people. Consequently, k anonymized data sets provide little information on users' mobility patterns, which makes it difficult to link multiple data points produced by the same user.

There are, however, many situations where we can improve our analytic methods by considering users' mobility patterns. For example, Draffic [5] provides a statistical analysis of people's movements in sightseeing areas so that hotels and souvenir shops can take effective measures to attract more visitors and provide them with better services. Similarly, a shopping mall manager could position various stores in the mall and thus conveniently match customers' shopping experience to their movement through the mall.

We, therefore, propose a new dynamic pseudonym scheme for constructing a location data set that retains users' path information while preserving their location privacy. Our basic approach is to exchange multiple users' pseudonyms only when they meet at the same location to eliminate the linkability of their pseudonyms before and after that exchange. We believe that such a dynamic pseudonym approach is effective since many people move through hub locations (e.g., a train station near sightseeing spots) where many people meet [6], as shown in Figure I. Our privacy metrics requires that, at a given time t, every user has a sufficient number of plausible paths heading towards K different locations.

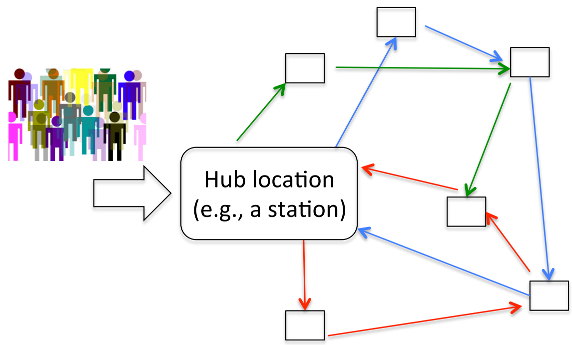

Figure 1. Location trajectories through location hubs. Arrows with the same color represent the movements of one person. Rectangles denote a location hub where many people meet.

To make this dynamic pseudonym-based scheme practical, we address the issue of multi-path inconsistencies among multiple users. Assuming that users' home locations are public knowledge available to an adversary [7], we find that not all pseudonym exchanges can be effective; the adversary can detect global inconsistencies among multiple plausible paths taken by different users. It is thus not trivial to decide whether a given data set is safely publishable. We, therefore develop a verification algorithm for determining whether it is possible to convert a given location data set into pseudonym-based data satisfying the (K, t)-privacy metrics. We prove both the soundness and completeness of our algorithm; the algorithm considers all the valid plausible paths of users while excluding all their invalid paths.

Although the simplicity of the original verification algorithm is convenient for proving its correctness, its running time is exponential. Therefore, we develop a polynomial-time version of an equivalent algorithm by reducing the user-pseudonym matching problems to a complete bipartite matching problem, which can be solved efficiently. We summarize our contributions in this paper as follows:

1) We develop a quantitative privacy metrics for pseudonym-based location data sets founded on the notion of possible paths.
2) We develop a polynomial-time algorithm that allows us to publish a pseudonym-based location data set in a privacy-preserving way.
3) We formally prove the correctness of the verification algorithm in terms of its soundness and completeness.

The rest of the paper is organized as follows. We introduce our system model for pseudonym-based location services in Section II and then define our privacy metrics in Section III. Next, we present a verification algorithm for a pseudonym-based location data set and prove its correctness in Section IV, and develop an equivalent algorithm running in polynomial time in Section V. Section VI discusses possible future work concerning the algorithms in Sections IV and V. We cover related work in Section VII and finally conclude the paper in Section VIII.

II. System model

Figure 2 shows our system model for pseudonym-based location systems. We assume that a mobile user u_i carrying a GPS-enabled mobile device periodically reports a triplet (u_i, l_k, t_k), which indicates that user u_i is at location l_k at time t_k. The pseudonym-based location server receives identifiable location data from multiple users, replaces the users' identities with pseudonyms, and provides location-based content providers, such as traffic monitoring applications, with location data that have pseudonyms.

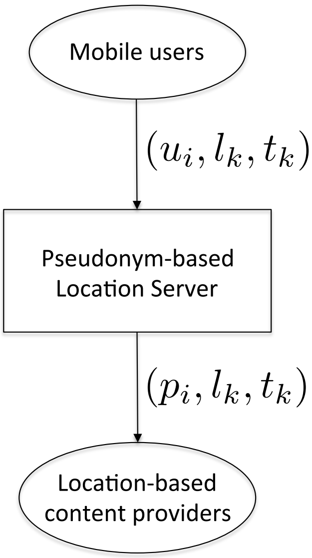

Figure 2. System model. The pseudonym-based location server replaces a user's identity u_i with a pseudonym p_i before releasing location data to content providers.

We first introduce the following four sets U, P, L, and T to define our system model.

U: a set of m mobile users such that $|U| = n$.
P: a set of m pseudonyms such that $|P| = n$.
L: a set of symbolic locations.
T: a set of timestamps $\{0, 1, \ldots, t^*\}$ where t^* is the last timestamp.

We next define the following four functions.

Definition 1 (User location function W_U): The location function $W_U : U \times T \rightarrow L$ returns the location l of user u at time t.

Definition 2 (Pseudonym location function W_P): The location function $W_P : P \times T \rightarrow L$ returns the location l of pseudonym p at time t.

Definition 3 (Pseudonym assignment function N): The pseudonym assignment function $N : U \times T \rightarrow P$ maps a user u at time t to a pseudonym p. We say that a user u owns a pseudonym p at time t if $N(u, t) = p$. For every time $t \in T$, the function $N_t(u) \equiv N(u, t)$ is a one-to-one function from U to P.
Note that $N(u, t) = p$ implies that $W_U(u, t) = W_P(p, t)$.

We next assume that each user $u_i \in U$ is associated with a home location l_i with the following home location function.

Definition 4 (Home location function H): The home location function $H : U \rightarrow L$ maps a user u_i to his home location l_i. Since we assume that each user has a different home location, function H is one-to-one.

We now define a pseudonym-based data set PL parameterized by the functions W_U and N as the following set of triplets:

$$PL = \{(p, l, t) \mid t \in T, u \in U, p = N(u, t), l = W_U(u, t)\}.$$

This data set represents the output from a pseudonym-based location server in Figure II. In this paper, we consider a malicious content provider who legitimately obtains a data set from the pseudonym-based location server and tries to violate the user's privacy corresponding to a certain pseudonym in the data set.

III. Pseudonym-based location privacy

To replace the user identity on a given moving path with the static pseudonym does not necessarily protect the user's location privacy. We take an approach that involves changing each user's pseudonym dynamically to prevent inference attacks using external knowledge about her home location.

A. Pseudonym exchanges

Each user u_i typically starts his moving path from his home $H(u_i)$ and finally returns there again. Therefore, if a user's home address is known to a malicious content provider, which is a common assumption in location privacy research [4], his moving path with the same pseudonym does not protect his location privacy; it is trivial to infer that the whole path belongs to the same user whose home address appears at both ends.

Therefore, it is necessary to change pseudonyms dynamically to prevent the above attack. The basic idea is to divide a whole path of the same user into multiple segments with different pseudonyms so that it is infeasible to link any neighboring segments. However, when a user

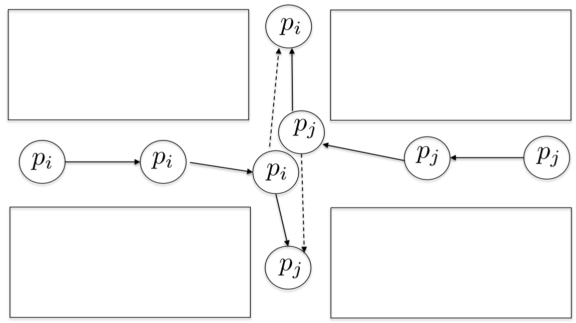

Figure 3. Example pseudonym exchange. Two users exchange their pseudonyms p_i and p_j at the intersection. The solid lines denote each user's actual path while the dotted lines denote an alternate possible path.

moves in an area where there are no other nearby users, it is straightforward to link two pseudonyms of the same user since we know that the user, who is subject to the laws of physics, cannot quickly jump to a distant place.

To address this issue, we adopt an approach in which we exchange multiple users' pseudonyms only when they meet at the same location, which is similar to that of using fresh pseudonyms in a mix zone [8]. Figure 3 shows an example of two users' exchanging their pseudonyms. Two users who own pseudonyms p_i and p_j, respectively, randomly exchange their pseudonyms when meeting at the intersection. Although the user who previously owned pseudonym p_i actually turns right at the corner, we consider that the alternate path turning left is also possible. The other user similarly has the two possible paths after passing the intersection.

To consider only such valid pseudonym exchanges, we put the following constraint on the pseudonym assignment function N. For every pair of two different users $u, u' \in U$ and time $t > 0$, if $N(u, t-1) = p$ and $N(u', t) = p$, then $W_U(u, t-1) = W_U(u', t-1)$ holds. Intuitively, this constraint implies that if a user u' receives another user u's pseudonym p at time t, users u and u' must have met at the same location at the previous time $t-1$.

B. Multi-path consistency

If we consider the possible paths of a single user, whenever the user meets another user, we can add a new branch as a possible segment of the path. However, we assume in this paper that every user starts from his home location and eventually returns there. Thus, we need to eliminate some possible branches if taking that direction makes it impossible for the user to return to his home location. Furthermore, even if one user u_i is able to return home along a possible path, another user u_j who exchanged her pseudonym with u_i might lose her possible route home.

We elaborate this multi-path consistency issue with the ladder model in Figure 4. The ladder model represents a pseudonym assignment function N in a graphical way abstracting away each user's physical movements. Figure 4 shows an example ladder model for three users u_1, u_2 and u_3. The model denotes each pseudonym p_i by a

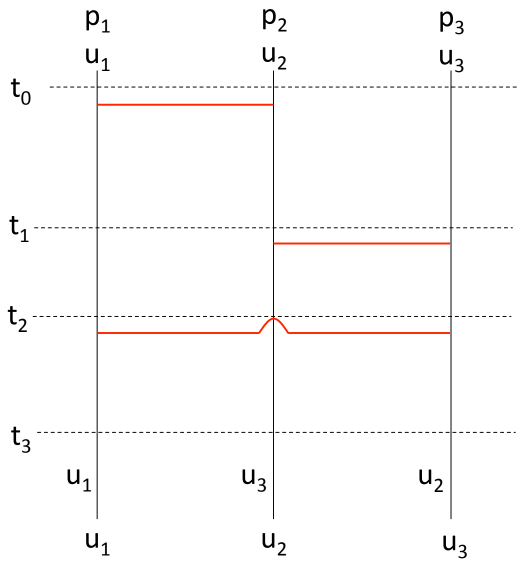

Figure 4. An example of time-changing pseudonym assignments based on the ladder model.

vertical line, and represents an encounter of multiple users associated with a different pseudonym by connecting their pseudonyms with a horizontal line. [1] Assuming that time passes vertically downward, we specify the sequential order of users' meetings by the positions of the horizontal lines.

Each pseudonym p_i is associated with a particular user at any given time t. In Figure 4, pseudonym p_1, p_2, and p_3 are associated with users u_1, u_2, and u_3, respectively, both at the start and end times t_0 and t_3; that is,

$$N(u_i, t_0) = N(u_i, t_3) = p_i \text{ for } i = 1, 2, 3.$$

This implies that for each user u_i,

$$W_U(u_i, t_0) = W_P(p_i, t_0) = H(u_i) = W_U(u_i, t_3) = W_P(p_i, t_3).$$

If we construct user u_1's possible time-changing pseudonym assignments by exchanging pseudonyms, we obtain the following sequences:

1) $p_1 \to p_1 \to p_1 \to p_1$
2) $p_1 \to p_2 \to p_2 \to p_2$
3) $p_1 \to p_2 \to p_3 \to p_3$
4) $p_1 \to p_2 \to p_3 \to p_1$

If we consider the requirements that user u_1 owns pseudonym p_1 at times t_0 and t_3, we must eliminate sequences (2) and (3) leaving (1) and (4) as possible sequences of pseudonym assignments. However, if we take the pseudonym sequence (4), users u_2 and u_3 are forced to take the pseudonym sequences $p_2 \to p_1 \to p_1 \to p_3$ and $p_3 \to p_3 \to p_2 \to p_2$, respectively, violating their endpoint requirements. Thus, it turns out to be impossible for user u_1 to take the pseudonym sequence (4) above.

We should therefore consider possible pseudonym sequences for multiple users simultaneously to ensure that

[1] Note that we can always convert an encounter of more than two users into a corresponding sequence of two-user encounters in our ladder model as we discuss in Section V-B.

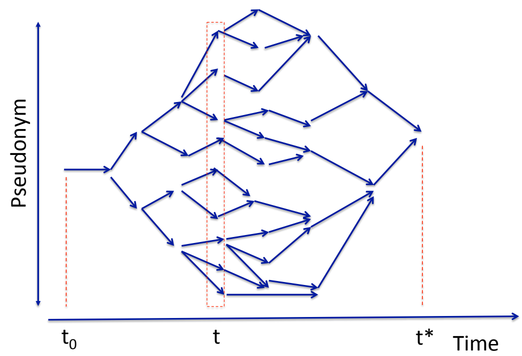

Figure 5. Concept of (K,t)-pseudonym location privacy. We assume that any pseudonym sequence following the arrows from left to right can be produced by a certain multi-path consistent pseudonym assignment function N.

the resulting pseudonym assignment function N satisfies the following multi-path consistency requirement.

Definition 5 (Multi-path consistent function N): We say that, for a given user location function W_U and a pseudonym location function W_P, a pseudonym assignment function N is multi-path consistent if

1) $\forall u, u' \in U, \forall t \in T > 0 : N(u, t-1) = p \wedge N(u', t) = p \Rightarrow W_U(u, t-1) = W_U(u', t-1)$,
2) $\forall u, u' \in U, \forall t \in T : N(u, t) \neq N(u', t)$, and
3) $\forall u \in U : W_P(p, t) = H(u) \Rightarrow N(u, t) = p$ for $t = 0, t^*$.

Note that the second condition should always hold since the pseudonym assignment function N in Definition 3 is one-to-one when we fix time $t \in T$.

C. (K,t)-pseudonym location privacy

We argue that the number of possible pseudonym sequences is not an appropriate privacy metrics for pseudonym-based location services. Consider the situation where two users move together taking the same moving path. If the two users possibly exchange their pseudonyms at each time, the result is an exponential number of possible pseudonym sequences with respect to the length of time. Therefore, we rather use the number of pseudonyms at a given time t on possible pseudonym sequences satisfying the multi-path consistency requirement as our location privacy metrics. Figure 5 shows such multiple pseudonym sequences of user u_i. There is only a single possible pseudonym at the initial time t_0 and the last time t^*. On the other hand, user u_i can take multiple pseudonyms in the middle of those sequences. If user u_i can take K or more pseudonyms at a given time t, we say that user u_i satisfies (K,t)-pseudonym location privacy.

We now formally define the notion of (K,t)-pseudonym location privacy as follows.

Definition 6 ((K,t)-pseudonym location privacy): Given a user u_i, we say that a location function W satisfies (K,t)-pseudonym location privacy if there exist K or more pseudonym assignment functions $N_0, N_1, \ldots, N_K$ that are multi-path consistent such that every $N_l(u_i, t)$ for $l = 0$ to K outputs a distinctive pseudonym.

Time	p_1	p_2	p_3
t_0	$\{u_1\}$	$\{u_2\}$	$\{u_3\}$
t_1	$\{u_1, u_2, u_3\}$	$\{u_1, u_2, u_3\}$	$\{u_1, u_2, u_3\}$
t_2	$\{u_1, u_2, u_3\}$	$\{u_1, u_2, u_3\}$	$\{u_1, u_2, u_3\}$
$t_3\ (= t^*)$	$\{u_1\}$	$\{u_2\}$	$\{u_3\}$

Figure 6. Example matrix A.

Time	Exchangeable pseudonyms
t_0	$\{p_1, p_2\}$
t_1	$\{p_2, p_3\}$
t_2	$\{p_1, p_3\}$

Figure 7. Example list AM.

IV. Verification of pseudonym-based location data sets

In this section, we describe an algorithm for determining whether a given data set satisfies the privacy metrics in Definition 6 and prove its correctness in terms of both soundness and completeness.

A. Verification algorithm

We present a privacy evaluation algorithm for computing how many possible pseudonyms each user u_i could have at a given time t. The algorithm takes two data structures $A[t,i]$ and $AM[t]$ as inputs. The matrix $A[t,i]$ contains a set of users who can possibly take a pseudonym p_i at time t. Initially, for all i, each field $A[t,i]$ contains the set of all users U except for $A[0,i]$ and $A[t^*,i]$, which only contains a single user. $A[0,i]$ and $A[t^*,i]$ contain users u_k and u_l respectively such that $W_P(p_i, 0) = H(u_k)$ and $W_P(p_i, t^*) = H(u_l)$. Figure 6 shows an example of matrix A where $A[0,i]$ and $A[t^*,i]$ contain a user u_i for $i = 1, 2, 3$. The list $AM[t]$ contains a set of pseudonyms that can be exchanged by their owner users at time t. The example AM in Figure 7 shows that pseudonyms p_1 and p_2 can be exchanged at time t_1.

Taking A and AM as inputs, the algorithm keeps updating the content of A propagating the constraints at both ends and outputs the final A, with which we can check how many pseudonyms a given user u_i takes at time t.

Algorithm 1 is the main program, which iteratively calls two functions O and I until matrix A cannot be updated any more. The function O sequentially narrows down the entries $A[t,i]$ at time t by computing all the possible mappings from pseudonyms to users at time t using the mapping information at time $t-1$. Function I performs this task in the reverse order.

Algorithm 2 shows how function O computes possible user-pseudonym mappings sequentially. Function O takes A and AM as inputs and updates A as follows.

The function *compPossibleMappings* in line 3 computes all the possible pseudonym-user mappings at time

Algorithm 1 Main program.

```
1: while 1 do
2:     prevA ← A
3:     A ← O(A, AM)
4:     A ← I(A, AM)
5:     if A = prevA then
6:         break;
7:     end if
8: end while
9: return A
```

Algorithm 2 Function O for computing possible user-pseudonym mappings sequentially.

```
1: for t = 1 → t* do
2:     seq ← ∅
3:     for all pseq ∈ compPossibleMappings(A, t − 1) do
4:         seq ← seq ∪ compCurrentSeqs(A, AM, t − 1, t, pseq)
5:     end for
6:     A ← replaceRow(A, t, seq)
7: end for
8: return A
```

$t-1$ from A as follows:

$$\begin{aligned}&compPossibleMappings(A, t)\\&= \{(u_1, \ldots, u_n) \mid \forall i : u_i \in A[t, i] \wedge \forall i, j : u_i \neq u_j\}.\end{aligned}$$

We represent such a mapping as a sequence of users. For example, mapping (u_1, u_2, u_3) means that u_1, u_2, and u_3 own pseudonyms p_1, p_2, and p_3, respectively. Line 3 stores such possible mapping in variable $psec$ and computes all possible pseudonym-user mappings by applying all the possible pseudonym exchanges specified in $AM[t-1]$. The variable seq on line 4 maintains all the user-pseudonym mappings at time t while iterating the for loop on each pseudonym-user mapping at time $t-1$. The function $compCurrentSeqs$ is formally defined as follows:

$$\begin{aligned}&compCurrentSeqs(A, AM, t_1, t_2, pseq)\\&= \{seq \mid seq \in exchangeable(pseq, AM[t_2]) \wedge \forall i : seq[i] \in A[t_2, i]\}\end{aligned}$$

where the function $exchangeable$ returns a list of possible pseudonym-user mappings derived from $pseq$ considering a list of exchangeable pseudonyms in list $AM[t_2]$. Finally, line 6 updates matrix A by replacing the ith row with a new row computed from the user-pseudonym mappings in seq using the function $replaceRow$. The outermost while loop iterates this operation sequentially from time $t = 1$ to t^*.

Similarly, Algorithm 3 shows how function I computes possible user-pseudonym mappings in the reverse order.

Algorithm 3 Function I for computing possible user-pseudonym mappings in the reverse order.

```
1: for t = t* → 1 do
2:     seq ← ∅
3:     for all nseq ∈ compPossibleMappings(A, t) do
4:         seq ← seq ∪ compCurrentSeqs(A, AM, t − 1, t − 1, nseq)
5:     end for
6:     A ← replaceRow(A, t − 1, seq)
7: end for
8: return A
```

Example: Consider the matrix A in Figure 6 again. At time t_0, only mapping $(u_1, u_2, u_3) \to (p_1, p_2, p_3)$ is possible. Therefore, the function $compPossibleMappings$ in line 3 returns the sequence (u_1, u_2, u_3), and that sequence is stored in variable $psec$. Next, the function $compCurrentSeqs$ computes the possible mappings at time t_1. If we look up $AM[0]$ in Figure 7, we learn that pseudonyms p_1 and p_2 are exchangeable at time t_1. Thus, we obtain two possible mappings (u_1, u_2, u_3) and (u_1, u_3, u_2). This implies that $A[1, 1] = \{u_1\}$, $A[1, 2] = \{u_2, u_3\}$, and $A[1, 3] = \{u_2, u_3\}$, and the function $replaceRow$ takes care of this task.

B. Completeness

We show that the verification algorithm in Section IV-A is complete in the sense that it does not to miss any valid assignment function N; that is, it maintains all the necessary elements in matrix A that are used to construct a possible path for some user $u_i \in U$

We first present formal definitions of notions such as possible path and possible mapping, which have already appeared in the previous sections. Then we establish the notion of a compatible list of possible paths satisfying matrix A, which corresponds to the multi-path consistent pseudonym assignment function N in Definition 5.

Let PL be a a pseudonym-based location data set with a user location function W_U and a pseudonym assignment function N. In the rest of this section we use a fixed user sequence $u_1, u_2, \ldots, u_n$ and a fixed pseudonym sequence $p_1, p_2, \ldots, p_n$ such that $u_i \neq u'_i$ and $p_i \neq p'_i$ if $i \neq i'$. Also we use $i(\_)$, $i'(\_)$, $j(\_)$ and $j'(\_)$ to denote any mapping from T to $\{1, \ldots, n\}$, and use $k(\_)$ to denote any permutation of $(1, \ldots, n)$.

Definition 7 (Possible path): We say that a sequence of elements $(p_{i(0)}, l_{j(0)}, 0)$, $(p_{i(1)}, l_{j(1)}, 1), \ldots, (p_{i(t^*)}, l_{j(t^*)}, t^*)$ in PL is a *possible* path if

$$\begin{aligned}\forall t \in T > 0 : p_{i(t-1)} &= N(u_{k(t-1)}, t-1)\\ \wedge\, p_{i(t)} &= N(u_{k(t)}, t)\\ \Rightarrow W_U(u_{k(t-1)}, t-1) &= W_U(u_{k(t)}, t-1).\end{aligned}$$

That is, a possible path satisfies the first condition in Definition 5.

Definition 8 (A pair of compatible paths): We say that a pair of paths $(p_{i(0)}, l_{j(0)}, 0)$, $(p_{i(1)}, l_{j(1)}, 1), \ldots, (p_{i(t^*)}, l_{j(t^*)}, t^*)$ and $(p_{i'(0)}, l_{j'(0)}, 0), (p_{i'(1)}, l_{j'(1)}, 1), \ldots, (p_{i'(t^*)}, l_{j'(t^*)}, t^*)$ is *compatible* if

$$\forall t \in T : (p_{i(t)}, l_{j(t)}, t) \neq (p_{i(t)}, l_{j(t)}, t).$$

Definition 9 (User of possible path): We say that a possible path $r = (p_{i(0)}, l_{j(0)}, 0)$, $(p_{i(1)}, l_{j(1)}, 1), \ldots, (p_{i(t^*)}, l_{j(t^*)}, t^*)$ satisfies A when there exists a user u such that $u \in A[i(t), t]$ for any $t \in T$. In such a case, we call u a possible user of r.

We are now ready to define the notion of a compatible list mentioned above.

Definition 10 (Compatible list): A *compatible list of possible paths for PL satisfying A* is a tuple $(r_1, r_2, \ldots, r_n)$ for which the following two conditions hold:

1) Each r_i is a possible path of PL satisfying A with possible user u_i.
2) A pair of paths r_i and r_j is compatible if $i \neq j$.

In the rest of this section, we identify a permutation $(u_{k(1)}, u_{k(2)}, \ldots, u_{k(n)})$ of the fixed user list $(u_1, u_2, \ldots, u_n)$ with the mapping $\{p_i \mapsto u_{k(i)} \mid i = 1, \ldots, n\}$, and we call such a permutation simply a *mapping*. We define a mapping at time t for a compatible list $s = (r_1, r_2, \ldots, r_n)$ as follows.

Definition 11 (mapping for a compatible list): Let $c = (r_1, r_2, \ldots, r_n)$ be a compatible list of possible paths for PL satisfying A. We say that a mapping $s = (u_{k(1)}, u_{k(2)}, \ldots, u_{k(n)})$ is a *mapping for* c *at* time $t \in T$ when the following condition holds:

$$\exists l \in L\ \forall i : \text{the } t\text{th element of } r_{k(i)} \text{ is } (p_i, l, t).$$

Clearly, a mapping at t is uniquely determined for any compatible list c, so we denote such a unique mapping by $c(t)$.

Definition 12 (Possibility of mapping): We say that a mapping $s = (u_{k(1)}, u_{k(2)}, \ldots, u_{k(n)})$ is *possible at* $t \in T$ *in* A when $u_{k(i)} \in A[t, i]$ for any $i = 1, \ldots, n$.

Definition 13 (Admissible permutation): Given a mapping $s = (u_{k(1)}, u_{k(2)}, \ldots, u_{k(n)})$ and a list of exchangeable pseudonyms AM, we say that another mapping s' is an admissible permutation of s with respect to $AM[t]$ if s' satisfies either of the following two conditions:

1) $s' = s$, or
2) $s' = swap(s, i, j)$ only if $\{p_i, p_j\} \in AM[t]$ where the function $swap(s, i, j)$ returns a permutation where s's ith and jth elements are swapped.

We now prove the following lemma concerning a time sequence of mappings for a given compatible list satisfying matrix A computed in Algorithm 1.

Lemma 1: We consider line 3 in Algorithm 1 where $A' = O(A, AM)$. If the following three conditions hold,

1) s' is a possible mapping at time $t-1$ in the new matrix A',
2) s is an admissible permutation of s' with respect to $AM[t-1]$,
3) s is a possible mapping at time t in the previous matrix A ,

then, s is a possible mapping at time t in A' .

A similar property holds with a time sequence of mappings in the reverse order.

Lemma 2: We consider line 4 in Algorithm 1 where $A' = I(A, AM)$. If the following three conditions hold,

1) s' is a mapping at time t in the new matrix A',
2) s is an admissible permutation of s' with respect to $AM[t-1]$,
3) s is a mapping at time $t-1$ in the previous matrix A ,

then, s is a possible mapping at time $t-1$ in A' .
We omit proofs for lemmas 1 and 2 since they are clear from the description of Algorithm 1.

We next prove that Algorithm 1 does not destroy any compatible list of possible paths present in the initial matrix A while removing elements from A by performing functions O and I iteratively.

Theorem 1 (Complete reduction of matrix A): If c is a compatible list of possible paths satisfying the initial matrix A of Algorithm 1, c remains to be a compatible list with respect to the a modified A at any step of the while loop in Algorithm 1.

Proof: We prove by contradiction. We assume that there is a step of the while loop where a compatible list $c = (r_1, \ldots, r_n)$ is destroyed. Without loss of generality, that list is destroyed when we execute function O to obtain new matrix $A' = O(A, AM)$; that is, c satisfies previous matrix A, but does not satisfy new matrix A'. Then, there exist a possible path $r_i = (p_{i(0)}, l_{j(0)}, 0), (p_{i(1)}, l_{j(1)}, 1), \ldots,$
$(p_{i(t^*)}, l_{j(t^*)}, t^*)$ and time t such that $u_i \in A[t, i(t)]$, but $u_i \notin A'[t, i(t)]$.

We consider such a situation with the smallest index i of a possible path r_i and the earliest time t. Then, the following statements must be true.

1) $c(t)$ is a possible mapping at t in matrix A. (The minimality of the iterations in the while loop)
2) $c(t-1)$ is a possible mapping at $t-1$ in matrix A'. (The minimality of time t)
3) $c(t-1)$ is an admissible permutation of $c(t)$ with respect to $AM[t-1]$.
4) $c(t)$ is not a possible mapping at t in A' by our assumption.

However, the fourth statement contradicts the conclusion in Lemma 1 and thus $c(t)$ must continue to be a possible mapping regarding A'. ■

C. Soundness

We show that Algorithm 1 does not to produce any pseudonym assignment function N that is not multi-path consistent. We show that any element in matrix A produced by Algorithm 1 is used as part of a possible path in a compatible list.

Lemma 3: Let $s_0, s_1, \ldots, s_{t^*}$ be a sequence of mappings where each s_t is possible at time t in matrix A. We assume that either of the two conditions hold.

1) A mapping s_t is an admissible permutation of s_{t-1} with respect to $AM[t-1]$.
2) A mapping s_{t-1} is an admissible permutation of s_t with respect to $AM[t-1]$.

Then, there exists a compatible list c satisfying matrix A such that $c(t) = s_t$ for each time $t \in T$.

Proof: We use $s_t(i)$ to denote the ith element of a mapping s. We also use $r_i(t)$ to denote the element of possible path r_i at time t. We define a list of possible paths $c = (r_1, \ldots, r_n)$ as follows:

$$r_k(t) = (p_i, l, t) \text{ iff } (p_i, l, t) \in PL \text{ and } s_t(i) = u_k.$$

First we show that $r_k = (p_{i(0)}, l_{j(0)}, 0), (p_{i(1)}, l_{j(1)}, 1), \ldots, (p_{i(t^*)}, l_{j(t^*)}, t^*)$ is a possible path with a possible user u_k satisfying A.

Since we assume that every mapping s_t for $t = 1, \ldots, t^*$ is possible at t in matrix A, $u_k \in A[t, i(t)]$ holds for all times in T. Let t be any time in T, and assume that $p_{i(t-1)} = N(u_{k(t-1)}, t-1)$ and that $p_{i(t)} = N(u_{k(t)}, t)$. We need to consider the following two cases.

If $p_{i(t-1)}$ is not exchangeable at t in AM, then $p_{i(t)} = p_{i(t-1)}$ by admissibility. Thus $u_{k(t-1)} = u_{k(t)}$ and so $W_U(u_{k(t-1)}) = W_U(u_{k(t)})$.

If $p_{i(t-1)}$ is exchangeable with, say, p' at t in AM, then either $p_{i(t)} = p_{i(t-1)}$ or $p_{i(t)} = p'$ holds. Let u' is be the user such that $N(u', t-1)$. Then, by the constraint on N, either $u_{k(t)} = u_{k(t-1)}$ or $u_{k(t)} = u'$ holds. Moreover, by exchangeability, $u_{k(t-1)}$ and u' must meet at the same location at time $t-1$. Thus, $W_U(u_{k(t-1)}) = W_U(u_{k(t)})$.

Therefore, r_k is a possible path with possible user u_k satisfying A. It is easy to see that every pair of distinct paths in c is compatible since $i \neq i'$ iff $st(i) \neq st(i')$, and that $c(t) = s_t$ by the construction. Therefore, a list $c = (r_1, \ldots, r_n)$ is a compatible list of possible paths satisfying A where $c(t) = s_t$ for every time t. ■

We now claim the soundness of the reduction processes on matrix A in Algorithm 1. We assume that A is the final matrix produced by Algorithm 1 below.

Lemma 4: For every tuple (u_j, t, i) where $u_j \in A[t, i]$, there exists a possible mapping s_t whose ith element is u_k.

Proof: When we execute function O for the last time, t^*th iteration of the for loop in Algorithm 2 updates $A[t^*, i]$ such that

$$A[t^*, i] = \{u'_i \mid (u'_1, \ldots, u'_i, \ldots, u'_n) \in seq\}$$

Since seq is a set of possible mappings at time t^*, there clearly exists a possible mapping s_{t^*} whose ith element is u_j. For time $t < t^*$, we can make the similar argument based on the operation semantics of function I. ■

Lemma 5: For every possible mapping s_t at time t, there exists a possible mapping s_{t-1}, which is an admissible permutation with respect to $AM[t-1]$.

Proof: The lemma is clearly true since $A = O(A, AM)$ and $A = I(A, AM)$ hold at the end of Algorithm 1. ■

Lemma 6: For every possible mapping s_t at time t, there exists a possible mapping s_{t+1}, which is an admissible permutation with respect to $AM[t]$.

Proof: A symmetrical argument for the proof of Lemma 5 holds. ■

Lemma 7: If s_t is a possible mapping at time t with respect to matrix A, then there exists a compatible list c such that $c(t) = s_t$.

Proof: By Lemmas 5, 6, and 3. ■

Theorem 2: For every tuple (u_j, t, i) where $u_j \in A[t, i]$, there exists a compatible list $c = (r_1, \ldots, r_n)$ where a possible path r_j contains an element (p_i, l, t) for some location l.

Proof: By Lemmas 4 and 7. ■

V. Polynomial-time equivalent algorithms

The time complexity of Algorithm 1 in Section IV is exponential in the worst case since both functions O and I perform iterations over permutations of all users. Fortunately, we can convert the original functions O and I into equivalent functions that run in polynomial time, and thus the main algorithm (i.e., Algorithm 1) also runs in polynomial time. In this section, we show an polynomial-time equivalent algorithm. We develop alternate versions of functions O and I and conduct a time complexity analysis of the new version of the algorithm.

A. Reduction to a bipartite matching problem

The main idea is to compute the current row of matrix A from the previous row (e.g., updating a row at $t-1$ row from row at t for function I) in both functions O and I by performing set intersection operations rather than examining every possible permutation at a time in each iteration of the for loop. For instance, consider example matrix A with the following two rows at times $t-1$ and t below.

T	p_1	p_2	p_3
$t-1$	$\{u_1, u_2, u_3\}$	$\{u_1, u_2, u_3\}$	$\{u_1, u_3\}$
t	$\{u_1, u_3\}$	$\{u_2\}$	$\{u_1, u_3\}$

We assume that this is a snapshot before updating the row at time $t-1$ using function I and that $\{p_2, p_3\} \in AM[t-1]$. We update the row at time $t-1$ as follows:

$$A[t-1, 1] \leftarrow A[t-1, 1] \cap A[t, 1],$$
$$A[t-1, 2] \leftarrow (A[t-1, 2] \cap (A[t, 2]) \cup (A[t-1, 2] \cap A[t, 3]),$$
$$A[t-1, 3] \leftarrow (A[t-1, 3] \cap (A[t, 3]) \cup (A[t-1, 3] \cap A[t, 2]).$$

Since there is no pseudonym exchange involving p_1, the same user should be associated with p_1 at times $t-1$ and t. Therefore, the user with p_1 should belong to the intersection $A[t-1, 1] \cap A[t, 1]$. Since pseudonyms p_2 and p_3 can be exchanged at time $t-1$, the user associated with pseudonym p_2 at time t could be associated either with p_2 or p_3 at time $t-1$. Therefore, the new $A[t-1, 2]$ should be the union of the two intersections $(A[t-1, 2] \cap (A[t, 2]) \cup (A[t-1, 2] \cap A[t, 3])$. These set operations update the row of matrix A at time t as follows:

T	p_1	p_2	p_3
$t-1$	$\{u_1, u_3\}$	$\{u_1, u_2, u_3\}$	$\{u_1, u_3\}$
t	$\{u_1, u_3\}$	$\{u_2\}$	$\{u_1, u_3\}$

However, performing such set operations is not guaranteed to produce the same results as the algorithms in Section IV and could thus violate the soundness property in Section IV-C. Note that if we perform the original function I on A instead, we obtain the following row at time $t-1$.

T	p_1	p_2	p_3
$t-1$	$\{u_1, u_3\}$	$\{u_2\}$	$\{u_1, u_3\}$
t	$\{u_1, u_3\}$	$\{u_2\}$	$\{u_1, u_3\}$

Note that $A[t-1, 2]$ only contains user u_2 since neither user u_1 nor u_3 in $A[t-1, 2]$ can be part of any possible mapping. Therefore, we also need to eliminate elements that cannot be part of possible mappings.

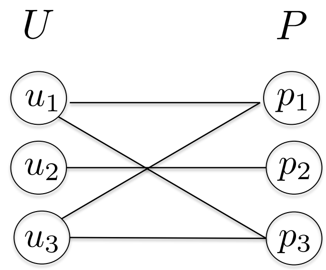

Figure 8. Example matching graph between two sets U and P.

To eliminate garbage elements that are not part of any possible mapping, we consider the problem of finding a complete pickup from a sequence of subsets as follows:

Definition 14 (Complete pickup): Given a sequence of sets $< A_1, \ldots, A_n >$ where each $A_i \subseteq U$, we say that a sequence of elements (i.e., users) $< a_1, \ldots, a_n >$ is a complete pickup if

1) $\forall i : a_i \in A_i$, and
2) $\forall i, j$ such that $i \neq j : a_i \neq a_j$.

The problem of finding a complete pickup can be reduced to a well-known bipartite matching problem as follows. We define a bipartite graph $G = (V, E)$ from a given a row of matrix A at time t as follows:

Definition 15 ((A,t)-bipartite graph): Given a row of matrix A at time t, we define a (A, t)-bipartite graph $G = (V, E)$ such that

1) $V = U \cup P$, and
2) $E = \{e_{ij} \mid u_i \in A[t, j]\}$ where e_{ij} is an edge between $u_i \in U$ and $p_j \in P$.

For example, if we consider the row at $t - 1$ below,

T	p_1	p_2	p_3
$t-1$	$\{u_1, u_3\}$	$\{u_2\}$	$\{u_1, u_3\}$

we obtain the bipartite graph shown in Figure 8. We now claim the equivalence between the two problems.

Proposition 1: Given a row of matrix A at time t, there exists a complete pickup in a sequence of sets $< A[t, 1], \ldots, A[t, n] >$ where each $A[t, i] \subseteq U$, if and only if the (A, t)-bipartite graph G in Definition 15 has a complete matching.
Thus, garbage elements in a row of matrix A at time t correspond to edges in the (A, t) bipartite graph that are not part of any complete matching.

Corollary 1: We say that a user u_i in $A[t, j]$ is a garbage if there is no complete matching with edge e_{ij} in (A, t)-bipartite graph G.

B. Revised functions O and I

We now describe an alternate algorithm of the function O, which runs in polynomial time. We omit the description of function I since we can derive it in a similar way. To simplify our description, we make the assumption that for every time t at most two users meet at the same location. We can represent a meeting of n users at the same location as a sequence of $(n(n+1)/2)$ two-user meetings at slightly different times. Such conversions include the number of times in T in the order of polynomial time. In the rest of this section, we use t^* to denote the last timestamp after performing such a conversion.

Algorithm 4 shows a polynomial-time version of the algorithm for function O. This function O considers two cases in the outermost for loop over every time $t \in T$. The first half between lines 4 and 11 covers the case where there is no pseudonym exchange at time $t-1$ whereas the second half between lines 12 and 19 covers the case where there is a pseudonym exchange at time $t - 1$. Lines 2 and 3 extract rows at time $t - 1$ and t respectively. Line 4 performs a set intersection between $A[t - 1, i]$ and $A[t, i]$ for each i and stores those intersections in the list *intSeq1*. The for loop between lines 5 and 11 examines every element a in the ith set in set sequence *intSeq1* and checks whether a is a member of any possible mapping in *intSeq1* by calling the function *checkExtensible*, which computes the maximum number of mappings in a bipartite graph. If a does not contribute to the construction of any possible mapping, line 8 removes it from the ith set *intSeq1*[1]. In the second case, we repeat the same procedure after swapping two sets in the row of A at $t - 1$ considering the possible pseudonym exchange mentioned in $AM[t-1]$. Line 20 updates the row of A at time t with the union of two resulting sequences *intSeq1* and *intSeq2*.

Algorithm 4 Polynomial-time function O.

```
1: for t = 1 → t* do
2:     prevSetSeq ← extractSetRow(A, t − 1)
3:     currentSetSeq ← extractSetRow(A, t)
4:     intSeq1 ← setSeqIntersection(prevSetSeq, currentSetSeq)
5:     for i = 0 → length(intSeq1) − 1 do
6:         for all a ∈ intSeq1[i] do
7:             if ¬checkExtensible(intSeq1, (a, i)) then
8:                 intSeq1[i] ← intSeq1[i] − {a}
9:             end if
10:        end for
11:    end for
12:    intSeq2 ←setSeqIntersection(exchange(prevSetSeq, AM, t −
       1), currentSetSeq)
13:    for i = 0 → length(intSeq2) − 1 do
14:        for all a ∈ intSeq2[i] do
15:            if ¬checkExtensible(intSeq2, (a, i)) then
16:                intSeq2[i] ← intSeq2[i] \ {a}
17:            end if
18:        end for
19:    end for
20:    A ← replaceSetRow(A, t, setSeqUnion(intSeq1, intSeq2))
21: end for
22: return A
```

We next describe the function *checkExtensible* in Algorithm 5, which verifies that an element a in the ith set in a set sequence *seq* is a complete pickup using the maximum cardinality bipartite matching algorithm (e.g., Dinic's algorithm in [9]). Lines 1 to 5 remove element a from all the sets in *seq* except from the ith set since we are only interested in maximum cardinality matching where element a in the ith set is used. The function *maxBipartiteMatching* in line 6 computes the maximum number of matchings in the modified set sequence *seq*. If there exists a maximum cardinality matching in *seq*, line 7 returns a True value; otherwise, line 9 returns a False value.

We finally claim that Algorithm 1 combined with the alternate version of functions I and O runs in polynomial

Algorithm 5 Function $checkExtensible$. INPUT: seq: a sequence of user sets; (a, i): an element a in the ith set.

```
for k = 0 → length(seq) − 1 do
    if k ≠ i then
        seq[k] ← seq[k] \ {a}
    end if
end for
if maxBipartiteMatching(seq) = length(seq) then
    return True
else
    return False
end if
```

time.

Theorem 3: The time complexity of Algorithm 1 with the alternate versions of functions I and O is $\Theta(n^8 * t^{*2})$

Proof: The while loop in Algorithm 1 iterates at most $n^2 * t^*$ times, which is the maximum number of elements in matrix A. Note that each field of A is a subset of all users U. Since every iteration must remove some element from A, the maximum number of iterations is bounded by the initial number of elements in A. The inner for loops in functions O and I iterate over every element in the current row of A performing the bipartite matching algorithm whose running time is $\Theta(n^4)$. Since that inner loop is iterated t^* times, the running time of O and I is $\Theta(n^2 * n^4 * t^*) = \Theta(n^6 * t^*)$. Thus, the total running time is $\Theta(n^8 * t^{*2})$. ■

VI. Discussion and Future work

There are a few possible extensions of the algorithm. First, an adversary might know that some location in the middle of a user's path is associated with a particular user. For example, the adversary might know a user's daytime office location. We can handle such additional external knowledge of an adversary with a minor modification of the algorithm. We just need to define an initial matrix A where some elements in A corresponding to known intermediate locations contain a single user. Second, it is desirable to keep longer path segments in a data set as long as that set preserves given privacy metrics. We plan to extend the current algorithm so that it determines the minimum number of items in an array AM that are needed to achieve given privacy metrics. Third, we would like to consider a realistic, weaker assumption, namely that an adversary only obtains a *partial* data set, which does not users' all path information. We expect that there is a better strategy for disguising the users' actual paths while satisfying the privacy metrics. Fourth, the problem setting in this paper is rather *possibilistic*, that is, we are interested in whether or not a user can hold a pseudonym at a given time. We expect to extend our results to more *probabilistic* settings, which will enable us to compute, for example, the probability that a privacy violation occurs.

VII. Related work

Several researchers [10], [11], [12], [13], [14] have proposed fine-grained access-control schemes based on rules for protecting location privacy in pervasive environments. Here, their focus is to provide a flexible policy language for protecting identifiable location data of mobile users. Hengartner [10] supports access-control policies considering the granularity of location information and time intervals. Myles [13] provides an XML-based authorization language for defining privacy policies that protect users' location information. Users must trust a set of validators that collect context information and make authorization decisions. Those schemes allow a user to define fine-grained access-control policies. Apu [12] provides users with an intuitive way of defining access control policies, which represent physical boundaries surrounding the users. However, no previous scheme has considered the issue of inference based on the mobility patterns of users.

Location privacy has been throughly studied in the context of the anonymization and obfuscation of location data (See [15] for a comprehensive survey). The focus of research in this area is to ensure that no anonymized and/or obfuscated data is associated with an individual. For example, Gruteser [4] proposes a scheme that changes the granularity of location information to ensure that each location contains at least k users (i.e., k-anonymity).

Using pseudonyms is a promising way to make location data unlinkable to a particular user. Beresford and Stajano [8] were the first to discuss the idea of dynamically changing pseudonyms in a mix zone where multiple people meet, in order to prevent an adversary from linking two pseudonyms of the same user. However, they only consider the situation where an adversary has just a local view of users' movements and observes pseudonyms of entering or leaving the same mix zone. Hoh and Gruteser [16] present a path perturbation algorithm that adds noises to original location data so that each user can construct alternate possible paths by exchanging his pseudonym with those of other users when they meet at the same place. However, their scheme does not consider an adversary's external knowledge that can associate each user with a particular home location, as we assume in this paper. On the other hand, our scheme does not add noises to location data to increase the number of points where multiple users meet. Instead, our algorithm computes all the combinations of users' valid alternate routes that satisfy the home locution constraints.

Buttyán et al. [17] studied the effectiveness of changing pseudonyms in the context of vehicular networks. They evaluated the linkability of consecutive pseudonyms assuming an adversary who can monitor the location traces of vehicles at a limited number of places. We are more concerned with the indistinguishability of a user's *global* paths rather the unlinkability of pseudonyms in *local* areas. Moreover, their adversary model is different from ours in that the adversary in our model can obtain location data with pseudonyms at any place although the adversary cannot physically see the movements of users in any limited area.

VIII. Conclusions

In this paper, we presented a dynamic pseudonym scheme for constructing confusing paths of mobile users to protect their location privacy. We introduced a formal definition of location privacy based on pseudonyms and showed a polynomial-time verification algorithm for determining whether each user in a given location data set has a sufficient number of possible paths to disguise his/her true movements. We provided proofs for both the soundness and completeness of the algorithm.

ACKNOWLEDGMENTS

This research is supported by the Strategic Joint Research Grant for NTT and Research Organization of Information and Systems (ROIS) and by the Grants-in-Aid for Scientific Research C, 11013869, of Japan Society for the Promotion of Science.

REFERENCES

[1] "Google maps," http://maps.google.com/. [Online]. Available: http://maps.google.com/

[2] T. Seike, H. Mimaki, Y. Hara, R. Odawara, T. Nagata, and M. Terada, "Research on the applicability of "mobile spatial statistics" for enhanced urban planning," *Journal of the City Planning Institute of Japan*, vol. 46, no. 3, pp. 451–456, 2011.

[3] D. Anthony, T. Henderson, and D. Kotz, "Privacy in location-aware computing environments," *IEEE Pervasive Computing*, vol. 6, no. 4, pp. 64–72, 2007.

[4] M. Gruteser and D. Grunwald, "Anonymous usage of location-based services through spatial and temporal cloaking," in *Proceedings of Mobisys 2003: The First International Conference on Mobile Systems, Applications, and Services*. San Francisco, CA: USENIX Associations, May 2003. [Online]. Available: http://www.usenix.org/events/mobisys03/tech/gruteser.html

[5] "Dentsu draffic," http://hd.lis-data.com/draffic/. [Online]. Available: http://hd.lis-data.com/draffic/

[6] M. C. Gonzalez, C. A. Hidalgo, and A.-L. Barabasi, "Understanding individual human mobility patterns," *Nature*, vol. 453, no. 7196, pp. 779–782, June 2008.

[7] J. Krumm, "Inference attacks on location tracks," in *Proceedings of the 5th international conference on Pervasive computing*, ser. PERVASIVE'07. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 127–143. [Online]. Available: http://dl.acm.org/citation.cfm?id=1758156.1758167

[8] A. R. Beresford and F. Stajano, "Location Privacy in Pervasive Computing," vol. 2, no. 1, pp. 46–55, January-March 2003. [Online]. Available: http://www.computer.org/pervasive/pc2003/b1046abs.htm

[9] R. E. Tarjan, *Data structures and network algorithms*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 1983.

[10] U. Hengartner and P. Steenkiste, "Access control to people location information," *ACM Transactions on Information and System Security (TISSEC)*, vol. 8, no. 4, pp. 424–456, 2005.

[11] J. I. Hong and J. A. Landay, "An architecture for privacy-sensitive ubiquitous computing," in *Proceedings of the 2nd international conference on Mobile systems, applications, and services (MobiSys)*. New York, NY, USA: ACM, 2004, pp. 177–189.

[12] A. Kapadia, T. Henderson, J. J. Fielding, and D. Kotz, "Virtual Walls: Protecting Digital Privacy in Pervasive Environments," in *Proceedings of the Fifth International Conference on Pervasive Computing (Pervasive)*, ser. LNCS, vol. 4480. Springer-Verlag, May 2007, pp. 162–179.

[13] G. Myles, A. Friday, and N. Davies, "Preserving privacy in environments with location-based applications," *IEEE Pervasive Computing*, vol. 2, no. 1, pp. 56–64, January-March 2003. [Online]. Available: http://www.computer.org/pervasive/pc2003/b1056abs.htm

[14] V. Sacramento, M. Endler, and C. de Souza, "A privacy service for location-based collaboration among mobile users," *Journal of the Brazilian Computer Society*, vol. 14, no. 4, pp. 41–57, 2008.

[15] J. Krumm, "A survey of computational location privacy," *Personal Ubiquitous Computing*, vol. 13, no. 6, pp. 391–399, 2009.

[16] B. Hoh and M. Gruteser, "Protecting location privacy through path confusion," in *Security and Privacy for Emerging Areas in Communications Networks, 2005. SecureComm 2005. First International Conference on*, sept. 2005, pp. 194 – 205.

[17] L. Buttyán, T. Holczer, and I. Vajda, "On the effectiveness of changing pseudonyms to provide location privacy in vanets," in *Proceedings of the 4th European conference on Security and privacy in ad-hoc and sensor networks*, ser. ESAS'07. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 129–141. [Online]. Available: http://dl.acm.org/citation.cfm?id=1784404.1784417