Track Me If You Can: On the Effectiveness of Context-based Identifier Changes in Deployed Mobile Networks

Laurent Bindschaedler[1,*] Murtuza Jadliwala[2,*] Igor Bilogrevic[1] Imad Aad[3]
Philip Ginzboorg[3] Valtteri Niemi[3]
Jean-Pierre Hubaux[1]
[1]EPFL, Switzerland. [2]Wichita State University, USA. [3]Nokia Research Center.

E-mail: firstname.lastname@{[1]epfl.ch, [2]wichita.edu, [3]nokia.com}

Abstract

Location privacy is a major concern in an increasingly connected and highly pervasive network of mobile users. Novel location-based applications and device-to-device services (on these mobile devices) are gaining popularity, but at the same time, these services allow curious service providers and eavesdroppers to track users and their movements. Earlier research efforts on location-privacy preservation, which were mostly based on identifier-change mechanisms in spatio-temporal de-correlation regions called mix-zones*, show that coordinated identifier-change techniques are reasonably effective in a simulation setting, although some smart attacks are still possible. However, a thorough analysis of these mechanisms that takes into consideration communication patterns and mobility from a real-life deployment is missing from these results. In this paper, we evaluate in a real-life setting the effectiveness of standard mix-zone-based privacy protection mechanisms against probabilistic tracking attacks. Our experiments involved 80 volunteers carrying smartphones for 4 months and being constantly eavesdropped on an adversarial mesh network of standard wireless Access Points (APs). To the best of our knowledge, this is the first study that provides empirical evidence about the effectiveness of mix-zone-based privacy-preserving mechanisms against practical adversaries in upcoming wireless and mobile systems.*

1. Introduction

Over the last few years, smartphones and mobile devices have replaced traditional personal computers as the main means to access web and other context-based services. A wide variety of applications are available on these devices, including applications for friend-finding [1], dating [2, 3], micro-blogging [27], localized advertisement [16], gaming, entertainment [49] and localized social networking [4, 7]; they take advantage of both infrastructure-based and device-to-device communications. Novel data-sharing applications have also emerged in societies that enforce strict censorship and data-regulation. For example, device-to-device messaging applications are being used in Iran to exchange sexually-explicit messages [39] and by anti-governmental insurgents in order to coordinate their actions [22]. New infrastructure-less systems for wireless device-to-device communications have been recently announced by Nokia [43], Qualcomm [17], Peep Wireless [30], and NEC [46].

In such pervasive communication systems, privacy, specifically location privacy, is always a critical issue. Eavesdroppers can constantly track users based on device identifiers in the broadcasted wireless messages. This information can be used by curious service providers in order to de-anonymize users and their availabilities [29], to track their preferences [26], to identify their social networks [23], or it can be used by malicious parties for nefarious activities [5, 20]. Infrastructure owners and third-party service providers who track user communications and location information in order to improve the offered service can inadvertently harm users' privacy if the collected data is leaked to unauthorized parties or improperly shared with corporate partners.

The location privacy problem in pervasive and mobile communication networks has recently received much attention. Inspired by Chaum's seminal work on mix networks [14, 15], Beresford and Stajano [9, 10] propose the concept of *mix-zones*, which are spatio-temporally defined regions where users can mix [41] or change [34, 13] their device identifiers, such as IP and MAC addresses. Application layer identities are generally encrypted by using session keys and are not visible. While in the mix-zone, users remain silent after this identifier change operation. They re-

*Equally contributing authors; ordered alphabetically. Murtuza was with EPFL when this work was accomplished.

sume communication with a new identifier (or pseudonym) after exiting the mix-zone. By providing de-correlation between users and their device identifiers, mix-zones make it difficult for an adversary to continuously track users simply by linking messages containing the same identifier. In cellular networks, location privacy is supported by changing a subscriber's Temporary Mobile Subscriber Identity, or TMSI [21], upon moving to a new geographical area.

The effectiveness of mix-zone-based identifier-change mechanisms has been studied in some of the recent research efforts [12, 28, 48], where authors proposed new location-privacy attacks in such systems. Buttyán et al. [13] and Freudiger et al. [24] focus on mix-zone schemes that consider specific network characteristics, whereas, Palanisamy et al. [42] propose schemes that guarantee a lower-bound on the level of achieved anonymity. Freudiger et al. [25] and Jadliwala et al. [36] address the problem of optimal mix-zone deployment by minimizing the probabilistic advantage of the adversary in tracking users. More recently, the mix-zone placement problem was also addressed from a game-theoretic perspective [35, 8] by deriving optimal attack and defense strategies for both eavesdroppers and users. However, a majority of these efforts consider only vehicular networking scenarios, which have significantly different network characteristics and mobility patterns compared to networking scenarios involving mobile humans. Moreover, due to the difficulty in conducting large-scale field trials, these results are based solely on simulations with rather restrictive parameters. Additionally, most of these studies assume a global passive adversary, which is a highly unrealistic assumption. Before this contribution, there did not exist a single field-study that evaluated context-based identifier-change mechanisms under a practical adversary model both on real mobile devices and in real communication scenarios. In our opinion, such a study is crucial in order to realistically evaluate the effectiveness of coordinated identifier-change mechanisms.

In this paper, we evaluate mix-zones and context-based identifier-change mechanisms by means of a real on-campus mobile network deployment consisting of 80 state-of-the-art Nokia N900 smartphones. These smartphones, carried and used by EPFL students and staff for a period of four months, were enabled with both standard infrastructure-based communications, such as cellular and WiFi, as well as a novel WiFi-based wireless peer-to-peer technology by Nokia, called Nokia Instant Community or NIC. We deployed an adversarial wireless mesh network of Access Points (APs) over a target region of the campus in order to eavesdrop on user communications in that region. In addition to developing a variety of custom applications to stimulate usage and participation, we implemented and deployed a context-based identifier-change service on all the deployed smartphones. This service performs a device identifier change operation based on the device context such as the number of neighborhood devices obtained from the wireless peer-to-peer channel. Setting up a real deployment is a challenging exercise with several non-trivial tasks such as volunteer recruitment, providing usage incentives, device configurations, application development and deploying regular updates. More details can be found in [6].

Given the data collected, during four month, from the adversarial mesh network, we first reconstruct the groundtruth of user movements and device identifiers. We then outline a Markov chain-based tracking framework for user tracking and propose two straightforward tracking strategies. By using well-established privacy metrics, we evaluate the effectiveness and cost of the deployed context-based identifier-change mechanism against the proposed tracking strategies for various adversarial strengths and parameters.

2. System Model

In this section, we outline the setup of our experimental mobile network and the deployed adversarial mesh network. We also describe the context-based identifier-change mechanism deployed on the mobile devices.

2.1. Mobile Network Model and Deployment

In order to conduct our experiments, we deployed a real mobile network testbed on the EPFL campus in Lausanne, Switzerland. The network model corresponding to the deployment is depicted in Fig. 1 (leftmost). We recruited 80 volunteers (mostly students and a few instructors) to participate in our four month-long experiments from March to June 2011. Each participant was equipped with, and was expected to use, a Nokia N900 smartphone for the duration of the experiment. In addition to accessing web services using the standard WLAN and cellular interfaces, participants also exchanged information with other co-located users by using an experimental wireless peer-to-peer messaging platform from Nokia called *Nokia Instant Community* or *NIC* [43]. NIC uses WiFi-based ad-hoc connections to seamlessly carry small-sized data between devices without any infrastructure requirement. The NIC wireless peer-to-peer interface not only provides a novel and inexpensive means for participants to seamlessly communicate with each other, but it is also useful in determining context such as device neighborhood. We stress that any other non-WiFi-based short-range wireless peer-to-peer communication interface, such as Bluetooth, could also be used for the purpose of context determination.

As mentioned earlier, participants used the NIC interface to exchange information with other colocated participants based on relationship, interests, affiliations and context. NIC communications are either *non-interactive*, where

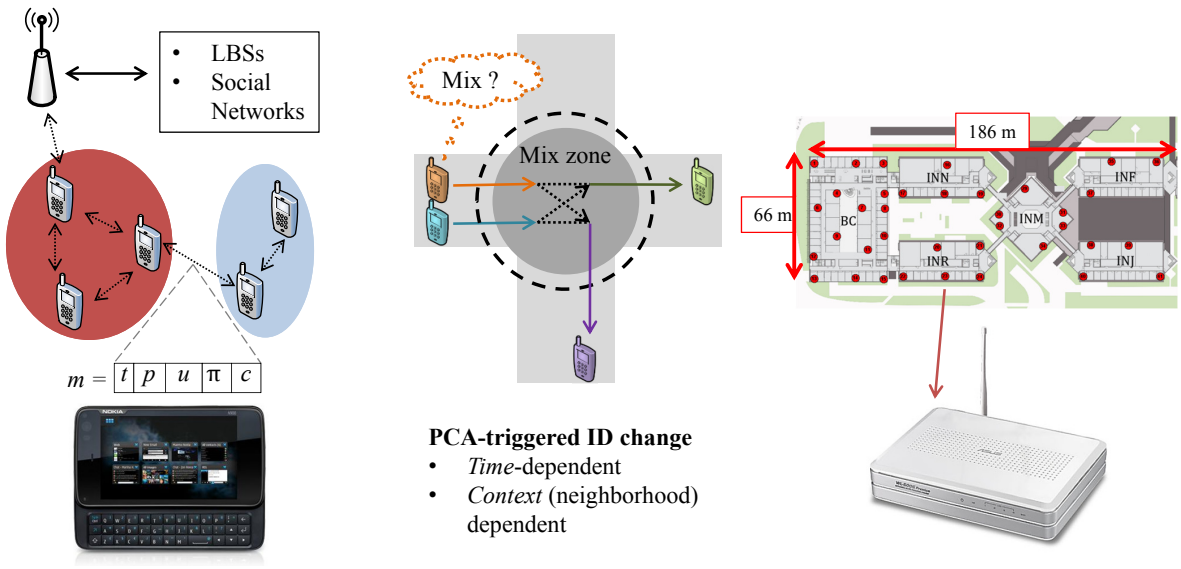

Figure 1. Left: System and network model with communities shown as shaded regions. Nokia N900 smart phone shown at bottom used in actual deployment. Middle: Mix-zone showing silent period and identifier changes. Right: Adversarial wireless mesh network of Asus routers.

users broadcast information without any specific request from others, or *interactive*, where users multi-cast relevant information only in response to specific queries. In order to stimulate information exchange, participants were recruited from closely-knitted groups, such as people in the same course or study group. To support both forms of communications, we developed seven custom NIC-based applications including a "*Classforum*" application for students to interact with the lecturer, a simple chat application, a restaurant application and a friend-finding application. These applications were either pre-loaded on the devices or remotely installed after deployment. Users could also dynamically create and organize themselves into communities using NIC. All users belonged to one large *public* community. *Private* communities could also be formed locally by users based on interest, preferences and affiliations. In adddition to this, participants also accessed Internet-based services, such as email, web browsing, etc., using these smartphones by connecting to the campus WiFi network. We note that, unlike Bluetooth, NIC-enabled smartphones seamlessly switch to the wireless peer-to-peer mode when NIC applications are running or other NIC devices appear in the neighborhood.

NIC messages are multi-hop and valid up to a certain maximum hop count. NIC messages carry limited data ($\approx$ 100 bytes) for efficiency and timing purposes (considering the short encounter period between devices). NIC implements an adaptive beaconing mechanism for *neighborhood discovery*. In order to prevent trivial linking of user messages from beacons, provisions are made in the beaconing mechanism, which allows a group of devices in the vicinity of each other to efficiently distribute the beaconing task uniformly within the group. Thus, the probability of the same device sending a beacon message regularly is low.

All NIC messages, similar to UDP/TCP segments, consist of (1) an unencrypted identifier or pseudonym (such as MAC and IP address) that belongs to the message originator and (2) the message itself in either encrypted or unencrypted form, depending on whether the message is private or public, respectively. If the message is not public, identifying information of the target user(s) or community is also included. Link-layer encryption [38, 50] could be used to encrypt MAC addresses, but the performance degradation due to the additional message size and cryptographic operations makes it infeasible for use in NIC. Each device runs (in the background) an identifier-change service, called Pseudonym-Change Algorithm (PCA), which conditionally changes the device MAC address. In this work, we assume that only the MAC address is changed, but the PCA can be easily extended to change application and network layer identifiers as well. For notational purposes, we denote an *actual message* sent by a user u as a five-tuple $m = (t, p, u, \pi, c)$, where t, p and π are the device time (in seconds), location (as a 2-D coordinate) and device identifier, respectively, when m was sent and c is the message content. Note that u is shown here only to associate a mes-

sage to a particular user and may not actually be sent with the message on the network. M is the set of all actual messages (sent by all users) and $t(m)$, $p(m)$, $\pi(m)$ refer to the time, position and identifier associated with m. A symbol table can be found in the Appendix.

2.2. Adversary Model and Deployment

In this work, we assume a passive adversary that eavesdrops on the messages sent by the devices and that is localized in a certain part of the network. We implement such an adversary by means of a mesh network of IEEE 802.11 wireless routers or APs (Asus WL-500gP APs running OpenWRT Linux), as shown in Fig. 1 (rightmost). This wireless mesh network of 37 APs is located on the same floor-level of six interconnected buildings, where there is high student activity because of the presence of various classrooms, study-areas and a cafeteria. Beyond this area, the adversary has no information on sent messages. The adversary has no direct access to any device, and all devices are assumed to be honest (i.e., non-colluding with the adversary).

Within the coverage area, the adversary is not be able to capture all possible messages due to hardware-related limitations, such as radio interference and hidden terminals, and AP software failures. The amount of data that the adversary can sniff or collect, i.e., adversary's strength, depends on the total number of APs used by her for sniffing. Each of these APs run a "tcpdump" background process to capture all communications on the specified channel and periodically upload the sniffed messages to a central server. The adversary is limited by the constraints of the cryptographic methods, i.e., she is not able to obtain/break the required cryptographic keys to decrypt private messages between devices. The adversary is not able to replay old messages or to inject or synthesize false messages into the network. Our adversary is weaker compared to the standard Dolev-Yao [19] model where the adversary is global, is able to overhear and intercept all messages, and is able to synthesize fake messages in the network. Our weaker, local eavesdropper adversary model is practical, given the current state of the art in monitoring technology, and it is sufficient for launching localized user-tracking attacks.

Let us represent a message observed by the adversary as a five-tuple $\hat{m} = (\hat{t}, \hat{\pi}, \hat{c}, \hat{o}, \hat{s})$, where $\hat{o}$ is the adversary AP observing the message, and $\hat{t}, \hat{\pi}, \hat{c}$ are the reception time (at the AP), device identifier or pseudonym and message content, respectively, and $\hat{s}$ is the received signal strength (in dB) measured by $\hat{o}$. Note that an observed message, obviously, does not contain the identity of the user u who sent the message, but only the device pseudonym $\hat{\pi} = \pi$. Let $\hat{M}$ be the set of all the messages observed by the adversary. The adversary attempts to link all pseudonyms (hence, the messages) belonging to a given user from the set of all observed pseudonymous messages, and thus to reconstruct the path taken by him (over the area observed by the adversary). We refer to this as a *reconstruction attack*, defined as follows:

Definition 1. *A reconstruction attack attempts to construct a temporal sequence or trace* $\tilde{r} = (\hat{\pi_0}, \hat{\pi_1}, \ldots, \hat{\pi_k})$ *of pseudonyms belonging to the same user, where* Π *is the set of all user pseudonyms and* $\hat{\pi_i} \in \Pi, \forall i$*. The corresponding true sequence, also called* ground-truth*, is denoted by* r*.*

The adversary's goal is to maximize the success of her reconstruction attack. We measure the success of the adversary by using well-established metrics, as described in Section 5.1. After all the traces belonging to a given user have been reconstructed, the adversary can attempt to deanonymize the actual identity of the user. Such re-identification attacks are beyond the scope of this paper; they could be carried out using several existing strategies in the literature [29, 26].

2.3. Pseudonym Change Algorithm (PCA)

In this work, our main aim is to evaluate context-based location-privacy protection mechanisms against passive adversaries as outlined in the earlier section. For this purpose, we implement a standard context-based identifier-change mechanism, referred to as Pseudonym Change Algorithm or PCA, which uses the concept of *mix-zones* [9, 10]. Mix-zones, as shown in Fig. 1 (middle), are dynamically formed regions (in our case, device neighborhoods) where users' devices stop transmitting and synchronously change their identifiers to achieve communication unlinkability. After exiting the mix-zone, the devices resume communication with new identifiers. It has been shown [10] that such coordinated identifier-changes provide sufficient spatio-temporal de-correlation between devices and their communications. But in order to be effective, identifier-changes should be reflected across all communication layers, including link, network and application layers. As application-layer identifiers can be encrypted and network addresses are dynamically assigned (at least in NIC), we focus on changing only the link layer or MAC address in our PCA.

We now briefly describe our PCA implementation (see Algorithm 1). Using a *context-check* timer t_c, the PCA periodically checks whether its current neighborhood is favorable (enough devices in the neighborhood) for an identifier-change operation. If there are enough devices in the neighborhood and if the total number (*num_context_changes*) of context-based identifier changes on the device, so far, is below some system-defined threshold Q_c, then the PCA initiates the MAC address change operation on the host device by calling the CHANGE_PSEUDONYM() function. It

Algorithm 1 Pseudonym Change Algorithm (PCA)

Define $Q_c \rightarrow$ context quota, $Q_f \rightarrow$ forced quota, $radio\_off\_time \rightarrow$ min. radio silence after identifier change, $change\_time\_threshold \rightarrow$ required min. time threshold and $neighbor\_threshold \rightarrow$ required min. neighbor threshold for identifier change.

on t_f expired **do** ▷ Forced timer expiration
  if $num\_forced\_changes < Q_f$ **then**
    BROADCAST($mix\_request$)
    CHANGE_PSEUDONYM( )
    RADIO_SILENCE($radio\_off\_time$)
    $num\_forced\_changes++$
  end if
end on

on t_c expired **do** ▷ Context timer expiration
  if current neighborhood $>$ $neighbor\_threshold$ **and** $num\_context\_changes < Q_c$ **then**
    BROADCAST($mix\_request$)
    CHANGE_PSEUDONYM( )
    RADIO_SILENCE($radio\_off\_time$)
    $num\_context\_changes++$
  end if
end on

on $mix\_request$ received **do** ▷ Incoming mixing request
  Let $d_t \leftarrow last\_change\_timestamp - current\_timestamp$
  if $d_t > change\_time\_threshold$ **and** $num\_context\_changes < Q_c$ **then**
    CHANGE_PSEUDONYM( )
    RADIO_SILENCE($radio\_off\_time$)
    $num\_context\_changes++$
  end if
end on

also initiates a *mix attempt* by sending a *mix-request* to the neighboring devices. After the identifier-change operation, PCA switches off the wireless interface for a random time period and increments the counter for context-based identifier changes. Random timers are used to prevent trivial timing-based linking attacks.

Upon receiving a mix-request, the neighboring devices follow a similar procedure for changing their MAC addresses; they perform an additional check before the identifier-change operation in order to prevent frequent changes that are ineffective and expensive. If the identifier-changes were solely based on device context, it would be possible that devices do not have a favorable context or neighborhood for a long period of time, thus making them trivially trackable. In order to overcome this issue, PCA makes use of another timer, called the *forced-change* timer or t_f. When t_f expires, the device, independently of its neighborhood, changes its MAC and enters a radio silence period if the number of forced identifier changes ($num\_forced\_changes$) is below the system-defined forced-change threshold Q_f. In order to prevent trivial timing attacks, both t_f and t_c are not fixed values but are chosen from a normal distribution (with appropriately chosen mean and variation) independently by each device. Note that the thresholds Q_c and Q_f are maximum *daily* thresholds and the counters $num\_context\_changes$ and $num\_forced\_changes$ are reset *daily* at 00:00 on each device.

In order to evaluate PCA under different privacy settings, we select three sets of parameter values, as shown in Table 1. The first set focuses on *cost-effectiveness* by requiring a comparatively lower number of overall identifier-changes and a larger time-gap between changes, which leads to fewer mixing attempts. Due to relaxed neighborhood requirements per change, the probability that the mixing attempt will be successful for this set of parameters is high. The third set of parameters has provisions for a large number of identifier-changes and a smaller time-gap requirement between changes, thus representing a *privacy-sensitive* behavior. The higher neighborhood requirement per change, in this case, is aimed at improving the unlinkability per change, i.e., each change is more effective than in the earlier case. Finally, the second parameter set represents an *intermediate* behavior.

Before the actual deployment, we divide the devices into three *groups*, each running PCA with a different parameter setting. In order to achieve a uniform distribution of parameters across all devices, the device parameters were shuffled across the groups remotely, once a month. In order to determine an appropriate value for the neighborhood threshold, we refer to the analysis by Kiukkonen et al. [40] who studied real user mobility based on mobile phone data (GPS locations and Bluetooth encounters) of 200 users collected over a period of 2 years. The authors observed that the average probability of any device having a Bluetooth encounter with more than 3 other devices was less than 0.2. WiFi-based peer-to-peer and Bluetooth communications differ in many aspects, but as WiFi-based peer-to-peer networks, such as NIC, are not commonplace, we rely on the above results to select initial values for the neighborhood thresholds. As NIC devices, unlike Bluetooth, are context-sensitive and may be switched-on all the time, the selected threshold values are a bit pessimistic, but they serve as a good starting point. Due to the limited duration of our deployment, we could not test a large variety of parameter configurations in real experiments. Later, as discussed in Section 5.6, we will perform simulation experiments with other PCA parameters and configurations by using the mobility patterns and data flows from the real deployment.

Table 1. PCA parameter values used in the real deployment.

Parameter	Mean t_f	Mean t_c	$Q = Q_c + Q_f$	*change_time_threshold*	*neighbor_threshold*
Cost-effective	14400 sec	3600 sec	5	7200 sec	1
Intermediate	7200 sec	1200 sec	20	1800 sec	2
Privacy-sensitive	3600 sec	300 sec	60	600 sec	3

3. Data Collection and Processing

In order to accurately measure the adversary's success, it is important to compare the results of her tracking attacks to the *groundtruth* (or actual user traces); the groundtruth acts as a reference for evaluating the success of the tracking attacks. The adversary wants to be as close to the groundtruth as possible. In this work, our first task is to reconstruct the groundtruth by adding extra information in the observed data. We assume that this extra information will not be available to the adversary, and as a result she will not be able to reconstruct the groundtruth as outlined in Section 3.1. Nevertheless, the adversary has to perform some basic processing on the observed data, as outlined in Section 3.2, before using it in her tracking attacks.

3.1. Groundtruth Reconstruction

The groundtruth reconstruction step provides the *true* mapping between each observed message and the device that sent it. This mapping will be used later to verify the correctness of the mapping created by the adversary's tracking algorithm. The source MAC address in the messages cannot be used for groundtruth reconstruction because it is regularly updated by the PCA. Thus, in order to obtain the groundtruth, we need to include some *fixed* device identifier in the observed messages. We achieve this by embedding the device *International Mobile Equipment Identity (IMEI)* number in the messages generated by a few user applications. The IMEI contained in the messages from these applications can be used to match different MAC addresses to the same device. For instance, two sets of messages, each with a different MAC address, can be linked to the same device (or user) if there is at least one message containing the IMEI for each MAC used. Even messages that do not contain the IMEI, such as system messages, can be linked together if there is at least one IMEI containing message with the same MAC address. In this way, we are able to reconstruct approximately 99% of the groundtruth from the observed messages.

3.2. Adversary Post-Processing

The adversary faces two main challenges while processing the observed or sniffed data: First, for any message m there may exist multiple copies of observed messages $\hat{m}$ (from different APs or sniffing stations), possibly with different values of $\hat{t}$, $\hat{o}$ and $\hat{s}$ (but obviously, with identical $\hat{\pi}$ and $\hat{c}$). Second, the position of the original message m is not directly available from the observed messages $\hat{m}$; it must be derived from the position of the sniffing stations ($\hat{o}$) and the Received Signal Strength Indicator (RSSI) ($\hat{s}$). For any message m, let $\tilde{m}$ be the re-construction of m by the adversary from the observed messages $\hat{m}$. The objective of the adversary is to reconstruct $\tilde{m}$ as close to m as possible. In order to do this, she has to first gather all observed messages $\hat{m}$ corresponding to the same sent message m. We refer to such a collection of identical observed messages as an *event* that can be represented as a five-tuple $e = (\hat{\pi}, \hat{c}, \hat{T}, \hat{O})$, where $\hat{\pi}$ is the pseudonym the device used while sending the message with content $\hat{c}$, $\hat{T}$ is the set of time-stamps corresponding to the times observed in each of the $\hat{m}$ forming e and $\hat{O}$ is the set of sniffing-station identifiers and corresponding RSSIs associated with each $\hat{m}$ forming e. The set of all events is denoted by E. The construction of E, referred to as *event identification*, can be accomplished by the adversary by matching the pseudonyms ($\hat{\pi}$) and contents ($\hat{c}$) of observed messages. After E is constructed, the adversary can use each event e in E to reconstruct the corresponding $\tilde{m}$ by aggregating all the information in that event. In other words, given an event e, the reconstructed message $\tilde{m}$ can be represented as $\tilde{m}(e) = (u, \tilde{t}, \tilde{p}, \tilde{\pi}, \tilde{c})$, where u is the user that sent the message m, $\tilde{\pi} = \hat{\pi}(e)$, $\tilde{c} = \hat{c}(e)$ and $\tilde{t}, \tilde{p}$ are aggregated values of time and position from $\hat{T}$ and $\hat{O}$ in e. The user u is not known to the adversary after the aggregation phase and it can be determined by using re-identification techniques [29, 26] after executing the tracking algorithms. We now outline the post-processing that the adversary does to construct $\tilde{m}$ (except user information) for each message m from a set of observed messages $\hat{m}$.

3.2.1 Time-stamp Synchronization

In order to accurately aggregate and order time-stamps during event identification, the adversarial sniffing stations or APs should be synchronized with each other. In our deployment, the APs suffer from synchronization problems due to large clock drifts (sometimes in several minutes) and random clock resets (in some rare cases) to the Unix Epoch, i.e., *1970-1-1 00:00:00*. Due to the unavailability of NTP-based solutions on these APs, we employ a *best-effort* syn-

chronization technique. We first divide time into discrete intervals of two hours. For each interval, we construct a graph where the (observed) APs represent the vertices of the graph and there is an edge between two vertices if a clock skew is detected between them. The clock skew gives the edge weight. Then, by means of a standard all-pairs shortest path algorithm such as Floyd-Warshall algorithm, we build a set of *least cost paths* between all pairs of sniffing stations. By starting from the node with the smallest cost path to all other nodes, we correct the clock skews step-by-step, until all clocks are resynchronized. The Nokia N900s are time-synchronized by using the NTP implementation on the devices.

3.2.2 Event Identification

Event identification consists of aggregating identical messages observed by different sniffing stations at (or approximately) the same time. A naive approach that searches the entire message space is infeasible. We use the following two heuristics, which significantly reduce the search space, and thus the event identification time.

- *Temporal proximity-based search*: Typically, any message broadcast by a device arrives at the neighboring sniffing stations within a delay of a few milliseconds of each other. Thus, we define an appropriate time threshold for each observed message and limit our search for identical messages to only those messages that have reception times within this threshold.
- *Spatial proximity-based search*: Assuming that all sniffing stations have the same range, any message sent by a device can be seen by sniffing stations only within a certain distance from the device. As defining distance thresholds is not practical, we define a neighborhood for each sniffing station and limit our search to messages observed by the station and its neighborhood.

We further reduce the processing time for message comparison by comparing the message content hashes (obtained using a cryptographic hash function such as SHA-2) rather than the actual content. This improves the comparison speed because of the reduced length and absence of long common prefixes in the compared strings.

3.2.3 Time-stamp Aggregation

After the event identification step, the adversary needs to reconstruct individual messages $\tilde{m}$ from their corresponding events. For each such message, she needs to first estimate the time at which it may have been sent by the source device. For a reconstructed message $\tilde{m}$ of the original message m, we assume that the adversary chooses the time-stamp $\tilde{t}$ as $min(\hat{T})$, where $\hat{T}$ is the set of time-stamps of the corresponding event. In other words, she chooses the lowest observed time-stamp for each event because it will be closest to the time instant at which the message was actually sent. The goal of aggregating time-stamps in such a fashion is to obtain an accurate ordering for reconstructed messages.

3.2.4 Coordinate Mapping

After assigning time-stamps to each reconstructed message $\tilde{m}$, the adversary needs to estimate its position, i.e., the position of the device when the corresponding m was sent. The adversary does this by aggregating the sniffing station and RSSI information present in each observed message of the event. The location of devices (hence, the messages) and sniffing stations can be represented as two-dimensional coordinates (x, y), where $0 \leq x \in \mathbb{R} \leq 200$, $0 \leq y \in \mathbb{R} \leq 100$. Note that the message (or device) positions can be slightly beyond the $186m \times 66m$ sniffing station deployment area. As the adversarial sniffing network is deployed on the same floor level, we do not consider the third dimension. For each reconstructed message $\tilde{m}$, the adversary first computes its distance to each sniffing station (observing that message) from the corresponding RSSI value in the event by using a standard radio propagation model [47] (Eqn. 1).

$$P_r = P_t + 20\log\left(\frac{\lambda}{4\pi}\right) + 10n\log\left(\frac{1}{d}\right) \quad (1)$$

In (1), P_t and P_r are the transmitted and received power [dBm], λ is the wavelength of the radio signal [m], n is the path-loss exponent and d is the distance between the transmitter and receiver [m]. In our deployment, we have $P_t = 20dBm$, $\lambda = 0.125m$ and $n = 4.8$ (considered suitable for indoor environments).

In Algorithm 2, we outline the procedure for computing the message position once the distances have been estimated. There can be four cases, depending on the distinct number of RSSIs available for the event, which in turn depends on the number of distinct observations (by sniffing stations) per message. In case only one observation is available, the position of the reconstructed message is chosen as the location of the sniffing station observing the message. If two observations are available, then a position weighted on the distance to each sniffing station is chosen. In the case three or more distinct distances to non-collinear sniffing stations are available, a straightforward trilateration procedure [37] is used by choosing an appropriate bound on the distance-error .

4. Tracking Framework and Algorithms

We now outline our probabilistic tracking framework and two tracking algorithms within this framework.

Algorithm 2 Position Estimation

Input: $P[]$ is the list of positions of sniffing stations observing the message. $P[i] = (P_x[i], P_y[i])$ are the x and y coordinates of the i^{th} sniffing station observing the message

Input: $\tilde{D}[]$ is the list of distances (computed from signal strength indicator) from each sniffing station observing the message

Output: $\tilde{p}$, the estimated position of the message

$n = |P|$, the number of sniffing stations
$m = |\tilde{D}|$, the number of distances
if not $(n = m)$ OR $n = 0$ **then**
    return ERROR
else if $n = 1$ **then** ▷ Only sniffing station i observes the message
    return $P[i]$
else if $n = 2$ **then** ▷ Only sniffing stations i and j observe the message
    return $(P_x[i] + \frac{P_x[j] - P_x[i]}{(\tilde{D}[i] + \tilde{D}[j])/\tilde{D}[i]}, P_y[i] + \frac{P_y[j] - P_y[i]}{(\tilde{D}[i] + \tilde{D}[j])/\tilde{D}[i]})$
else if $n = 3$ **then** ▷ Only sniffing stations i, j and k observe the message
    return TRILATERATION($P[i]$, $\tilde{D}[i]$, $P[j]$, $\tilde{D}[j]$, $P[k]$, $\tilde{D}[k]$)
else
    Let l, m, n be the three sniffing stations in $P[]$ with the smallest distances
    return TRILATERATION($P[l]$, $\tilde{D}[l]$, $P[m]$, $\tilde{D}[m]$, $P[n]$, $\tilde{D}[n]$)
end if

4.1. Probabilistic Tracking Framework

We model the daily observations by the adversarial AP mesh network by using a Markov chain, where a state of the system is defined as the set of ordered observations with the same pseudonym and the transition probability gives the probability of going from one set of observations to the next. As the total number of observations is finite and the transitions probabilities are time-invariant (as discussed below) with a fixed probability distribution on the initial states, the proposed model can be characterized as a *finite state first order Markov chain*.

Let S be the state space and each state $s \in S$ be represented as a triple of the form $s = (\pi, e_{start}, e_{end})$, where $\pi \in \Pi$ is a pseudonym and e_{start}, e_{end} are temporally[1] the first and last event respectively using the pseudonym π. Here, $t_{start}(s) = \tilde{t}(e_{start}(s))$ and $t_{end}(s) = \tilde{t}(e_{end}(s))$ are the corresponding start and end times of the state s, respectively. A similar notation is used for the observers and received signal strengths of the state. The transition probability matrix is defined by a transition function $P : S \times S \rightarrow [0, 1]$ that satisfies the following two constraints: 1) *Validity*: $\sum_{s_j \in S} P(s_i, s_j) = 1 \; \forall s_i \in S$ and 2) *Time monotonicity*: $P(s_i, s_j) = 0 \; \forall s_i, s_j \in S$ such that $t_{end}(s_i) > t_{start}(s_j)$. Each entry in the transition probability matrix $P[i, j]$ represents how likely the adversary considers state s_j to follow s_i; it gives the likelihood of the adversary to observe events in s_j immediately after observing events in s_i for any user. The transition probability matrix is user-invariant, i.e., the adversary's transition function is the same for all users. The tracking success of the adversary depends on how well the transition function is defined and how close it is to the actual transitions. In this work, we estimate the transition function by using two heuristics that intuitively capture the likelihood of observing a set of events from some previous events.

1. *Common sniffing stations*: The probability that a state s' is the next state of state s is proportional to the number of common sniffing stations in the events observed in s and in s'. The higher the number of common sniffing stations between s and s', the higher the probability of transitioning from s to s'.

2. *Speed matching*: As each event has position and time information, a notion of *speed between two events* can be defined as the Euclidean distance between the (position of) two events divided by the time difference between the events. Then, according to our speed-matching heuristic, the probability that a state s' follows state s is $1 - |v(s) - v(s')|/v_{max}$, where $v(s)$ is the speed of state s and v_{max} is the maximum speed ($5m/s$ in our case).

The tracking framework and the transition probabilities described above are for a single user, given his initial state (or some probability distribution on it). In this case, the attacker wants to determine a single (most probable) path in the state space corresponding to the user in the initial state. It is also possible to track multiple users simultaneously using the above model, i.e., the attacker will attempt to find several non-overlapping paths in the state space, given the initial states of multiple users. As the transitions in this case are between *sets* of states or *multi-states* (one state for each user being tracked), the transition probability function needs to be accordingly modified. The transition probability of one multi-state to the next can be estimated by normalizing the sum of the individual transition probabilities in the multi-set. Specific details of the single user and multi-user transition probability matrix computations can be found in [11].

[1] An implicit assumption here is that no pseudonym is reused.

4.2. Tracking Strategies

We propose two straightforward adversarial tracking strategies based on the above tracking framework.

4.2.1 Locally Optimal Walk (L-WALK)

The L-WALK algorithm (Alg. 3) reconstructs the user trace by performing a locally-optimal walk in the state space. In other words, at each state (of the walk), starting from the initial state, the next state candidate with the highest probability is selected, until there are no further candidates. L-WALK produces an ordered sequence of states $T = (s_0, s_1, ..., s_k)$ $(k > 0)$ such that $P(s_i, s_{i+1}) > P(s_i, s') \; \forall \; s' \in S \setminus s_{i+1} \; \forall \; i = 0, 1, ..., k$.

Algorithm 3 Locally optimal walk (L-WALK)

Input: Initial state s_0, heuristic function P
Output: Trace $T = (s_0, ..., s_k)$

1: $T \leftarrow (s_0)$, $continue \leftarrow True$
2: **repeat**
3: Let c be the next state candidate with the highest probability
4: Append c to T
5: **until** no more candidates
6: **return** T

From Alg. 3, we can see that L-WALK iteratively builds a walk by selecting the probabilistically best next state at each iteration and stops when no more states are available. It is possible to improve L-WALK by using additional reference points along the walk. For instance, if the adversary is able to ascertain that some state s_j is associated to a given user, she can use s_j to better direct her search in cases where multiple next states are equally probable at a given state. Due to this additional knowledge, paths that do not pass through s_j can also be discarded by the adversary. More than one such reference point will further improve the accuracy of the adversary's walk.

4.2.2 Globally Optimal Walk (G-WALK)

The G-WALK algorithm (Alg. 4) reconstructs the user trace by performing a walk in the state space such that the probability over the entire walk is maximized over all walks. In other words, G-WALK produces an ordered sequence of states $T = (s_0, s_1, ..., s_k)$ $(k > 0)$ such that $\Pi_{i=0}^{k-1} P(s_i, s_{i+1}) > \Pi_{j=0}^{l-1} P(s'_j, s'_{j+1}) \; : \; \forall T' = (s'_0, s'_1, ..., s'_l)$ with $T' \neq T$ and $s_0 = s'_0$. Unlike L-WALK, G-WALK does not rely on locally optimal choices but makes a globally optimal choice.

An exhaustive search for a globally optimal path is infeasible due to the large number of paths in the state space. In order to obtain a fairly good approximation of the global optimum, we use a randomized search heuristic called Simulated Annealing (SA) [33]. In G-WALK, the current solution in each iteration is replaced by a random "nearby" solution chosen with a probability that depends both on the difference between the values of those solutions and a global parameter t, called the temperature, which is gradually decreased in each iteration. The "nearby" solution or path is chosen by the MUTATE function, which replaces one of the state transitions in the current solution by an alternate one. The temperature parameter enables the algorithm to escape local optima throughout the simulation, while slowly converging to a near-optimal solution.

Algorithm 4 Globally optimal walk (G-WALK)

Input: Initial state s_0, transition function P, maximum evaluation count c_{max}, initial temperature $t_{initial}$ and temperature decrease factor $k \in [0, 1]$
Output: Trace $T = (s_0, ..., s_k)$

$T \leftarrow$ L-WALK(s_0), $v \leftarrow \Pi_{i=0}^{k-1} P(s_i, s_{i+1})$ ▷ Initial solution and likelihood
$T_{best} \leftarrow T$, $v_{best} \leftarrow v$ ▷ Best solution and likelihood
$c \leftarrow 0$ ▷ Initial evaluation count
while $c < c_{max}$ **do**
  $T_{new} \leftarrow$ MUTATE(T), $v_{new} \leftarrow \Pi_{i=0}^{k-1} P(s_{new,i}, s_{new,i+1})$
  $t \leftarrow$ TEMPERATURE(c), $c \leftarrow c + 1$
  if RANDOM( ) $<$ ACCEPT_PROBABILITY(v, v_{new}, t) **then** ▷ Randomly move to a new state
    $T \leftarrow T_{new}$, $v \leftarrow v_{new}$
  end if
  if $v_{new} > v_{best}$ **then**
    $T_{best} \leftarrow T_{new}$, $v_{best} \leftarrow v_{new}$
  end if
end while
return T_{best}

function MUTATE(T)
  Let T' be a random mutation of T.
  return T'
end function

function TEMPERATURE(c)
  return $t_{initial} \cdot (k)^c$
end function

function ACCEPT_PROBABILITY(v, v_{new}, t)
  return $min(1, e^{(v_{new} - v)/t})$
end function

5. Empirical Results and Evaluation

We evaluate the success of the L-WALK and G-WALK algorithms in tracking users by using well-known metrics for location privacy.

5.1. Privacy Metrics

We use four metrics in our evaluation; each uniquely captures the extent to which users can be tracked.

5.1.1 Traceability Metrics (τ-metrics)

Traceability metrics, originally proposed by Hoh et al. [32], capture the extent to which the user can be tracked in time or distance. In our setting, we define two traceability metrics: the percentage of time τ_t and the percentage of space τ_d during which tracking is successful. Formally, these metrics are defined as follows:

Definition 2 (Traceability metrics)**.** *Let $\tilde{r} = (\tilde{\pi_0}, \tilde{\pi_1}, ..., \tilde{\pi_l})$ denote the reconstruction of user pseudonyms as obtained by a reconstruction attack and $r = (\pi_0, \pi_1, ..., \pi_k)$ be the real sequence (groundtruth) of pseudonyms. The time- and distance-traceability metrics are:*

$$\tau_t(r, \tilde{r}) = \frac{\sum_{i=0}^{min(k,l)} t_{match}(\pi_i, \tilde{\pi}_i)}{\sum_{i=0}^{k} t_{used}(\pi_i)}$$

$$\tau_d(r, \tilde{r}) = \frac{\sum_{i=0}^{min(k,l)} d_{match}(\pi_i, \tilde{\pi}_i)}{\sum_{i=0}^{k} d_{used}(\pi_i)}$$

where $t_{used}(\pi)$ and $d_{used}(\pi)$ are the actual time and distance that pseudonym π was used by the user being tracked, and $t_{match}(\pi, \tilde{\pi})$ and $d_{match}(\pi, \tilde{\pi})$ are the time and distance during which pseudonym $\tilde{\pi}$ is correctly matched to π in the reconstructed trace by the adversary for that user.

5.1.2 Uncertainty Metrics (u-metrics)

Uncertainty metrics, originally proposed by Diaz et al. [18], capture the uncertainty of the adversary to correctly predict the next pseudonym used by the user. We define three uncertainty metrics as follows.

Definition 3 (Uncertainty metrics)**.** *Let $\tilde{r} = (\tilde{\pi_0}, \tilde{\pi_1}, ..., \tilde{\pi_l})$ denote the reconstruction of user pseudonyms obtained by a reconstruction attack. Let $Y(\overline{\pi})$ be the random variable denoting the next possible pseudonym (selected by the adversary) of pseudonym $\overline{\pi}$ and $\mathcal{H}$ denote the entropy operator. Then, we have the following uncertainty metrics that give the average, maximum and minimum entropy for all pseudonym changes:*

$$u_{avg}(\tilde{r}) = \frac{1}{l} \cdot \sum_{i=0}^{l-1} \mathcal{H}(Y(\pi_i))$$

$$u_{max}(\tilde{r}) = \max\{\mathcal{H}(Y(\pi_i)) \mid \pi_i \in \tilde{r} \setminus \tilde{\pi}_l\}$$

$$u_{min}(\tilde{r}) = \min\{\mathcal{H}(Y(\pi_i)) \mid \pi_i \in \tilde{r} \setminus \tilde{\pi}_l\}$$

5.1.3 Traceability-Uncertainty Metrics (μ-metrics)

Traceability-Uncertainty metrics combine the advantages of the traceability and uncertainty metrics by capturing both the extent to which the user can be tracked, as well as the difficulty (or uncertainty) in tracking. They can be defined as follows.

Definition 4 (Traceability-uncertainty metric)**.** *Let $\tilde{r} = (\tilde{\pi_0}, \tilde{\pi_1}, ..., \tilde{\pi_l})$, $r = (\pi_0, \pi_1, ..., \pi_k)$, $Y(\overline{\pi})$ be defined as above. Let $\mathcal{H}_{max}(Y(\overline{\pi})) = \log_2 |Y(\overline{\pi})|$ denote the maximum achievable entropy for pseudonym $\overline{\pi}$. Then, the corresponding time- (μ_t) and distance-traceability-uncertainty (μ_d) metrics are:*

$$\mu_t(r, \tilde{r}) = \frac{\sum_{i=1}^{min(k,l)} \frac{\mathcal{H}(Y(\pi_{i-1}))}{\mathcal{H}_{max}(Y(\pi_{i-1}))} \cdot t_{match}(\pi_i, \tilde{\pi}_i)}{\sum_{i=1}^{min(k,l)} t_{used}(\pi_i)}$$

$$\mu_d(r, \tilde{r}) = \frac{\sum_{i=1}^{min(k,l)} \frac{\mathcal{H}(Y(\pi_{i-1}))}{\mathcal{H}_{max}(Y(\pi_{i-1}))} \cdot d_{match}(\pi_i, \tilde{\pi}_i)}{\sum_{i=1}^{min(k,l)} d_{used}(\pi_i)}$$

5.1.4 Clustering Metrics (c-metrics)

Clustering c-metrics, as proposed by Hoh et al. [31], capture the extent to which one user was confused with another in the context of multiple user tracking. We define the clustering error metric as a measure of the average distance-error between the reconstructed and real sequence of pseudonym over all users. We first define the clustering error for a single user as follows:

Definition 5 (Clustering error for single user)**.** *For any user u, let $\tilde{r} = (\tilde{\pi_0}, ..., \tilde{\pi_l})$ and $r = (\pi_0, ..., \pi_k)$ be defined as above and $p_{start}(\overline{\pi})$ be the position of the start event of the subtrace with pseudonym $\overline{\pi}$. Then, the clustering error c_u for user u is the average distance between the start events in the reconstruction and in the groundtruth:*

$$c_u(r, \tilde{r}) = \frac{1}{min(k, l) - 1} \cdot \sum_{i=0}^{min(k,l)} \|p_{start}(\tilde{\pi}_i) - p_{start}(\pi_i)\|$$

Then, the multi-user clustering error metric can be defined as follows.

Definition 6 (Clustering error metric)**.** *Let $\tilde{R}$ be the set of reconstructed sequences of user pseudonyms as obtained*

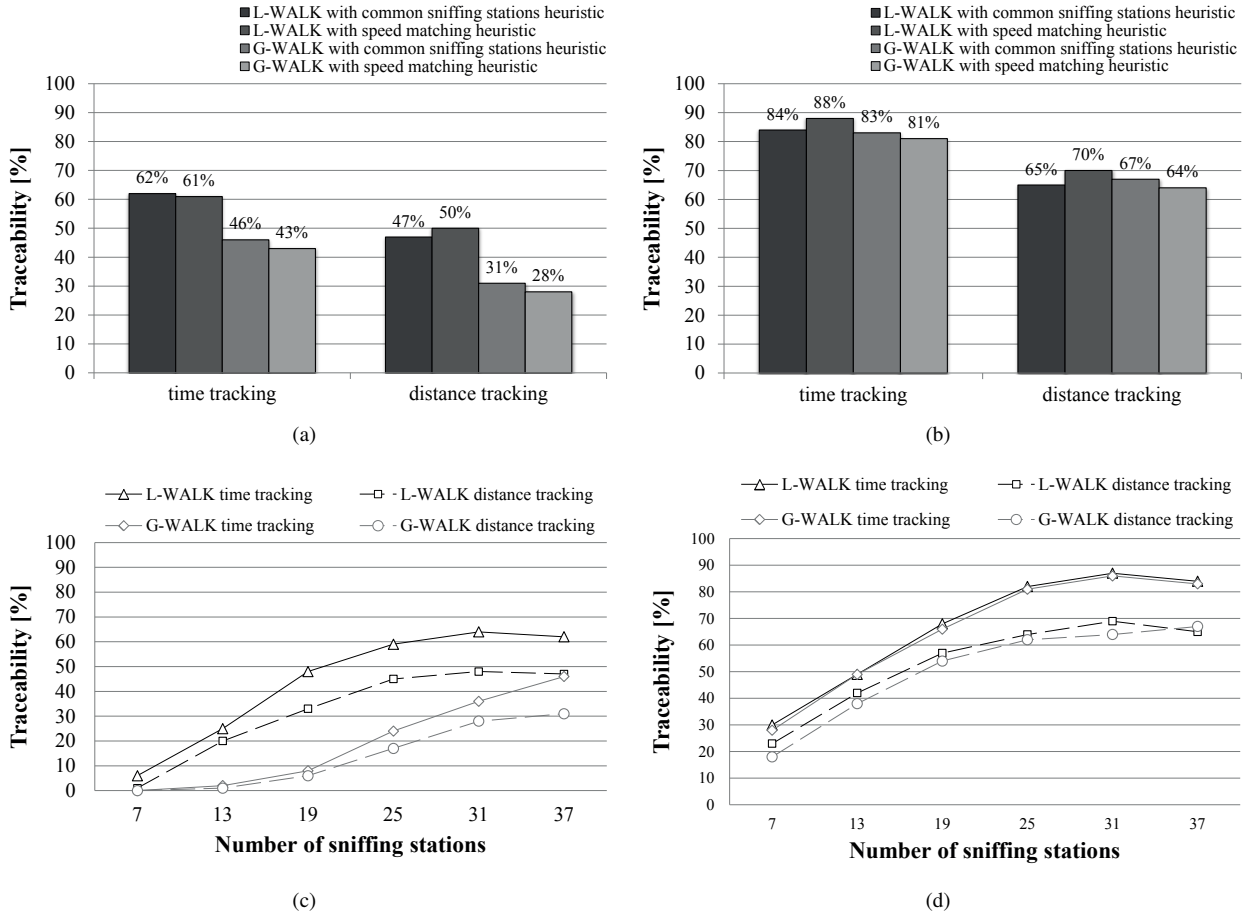

Figure 2. Traceability success: 2(a) single user tracking results; 2(b) multiple user tracking results. Adversary strengths using `L-WALK` and `G-WALK`: 2(c) single user; 2(d) multiple users.

by a multi-user reconstruction attack, R be the set of real sequences (groundtruth) of pseudonyms for the same users and U be the set of all users. Then, the the clustering error metric C is defined as the average clustering error over all users:

$$C(R,\tilde{R}) = \frac{1}{|U|} \cdot \sum_{u \in U, r \in R, \tilde{r} \in \tilde{R}} c_u(r,\tilde{r})$$

The evaluation of the proposed tracking algorithms with more sophisticated privacy metrics, such as the one by Shokri et al. [45, 44], is left as part of future work.

5.2. Results Overview

Fig. 2(a) shows the results for the single-user tracking using time and distance traceability metrics, which are averaged across all users and all tracking attempts over the entire 4-month duration. We can observe that the success ratio of `L-WALK` is around 60% with the time traceability metric and roughly 50% with the distance traceability metric. Also, both the speed and common station heuristics fared more or less equally well. The high success ratio of the simple `L-WALK` algorithm shows that the current PCA parameters (Table 1), althought realistic, do not perform very well in networking scenarios with slow or limited mobility, such as a network of smartphone users. In our setting, it is indeed rare to observe users move significantly during a mixing attempt. Consequently, even simple algorithms, such as `L-WALK`, are very effective in providing a correct ordering on candidates for the next state, despite the fact that transition probability estimates are not extremely precise.

In order to have an equivalent performance comparison of the `G-WALK` with `L-WALK` for the single user case, we need to slightly adapt `G-WALK`'s maximization criterion. In the single-user case, simply maximizing the product of probabilities for each complete walk is inherently prone to selecting shorter walks (as their products tend to be larger than those of longer walks). In our implementation, rather than simply maximizing the product of proba-

bilities, G-WALK selects the walk that maximizes the geometric average of its probabilities, i.e., $\sqrt[k]{\Pi_{i=0}^{k-1} P(s_i, s_{i+1})} > \sqrt[l]{\Pi_{i=0}^{l-1} P(s'_i, s'_{i+1})}$ $\forall T' = (s'_0, s'_1, ..., s'_l)$ with $T' \neq T, s'_i \in S$ $\forall\, i = 0, 1, ..., k$. Thus, by taking into account the length of each walk, we avoid the previous bias towards selecting shorter walks.

Experimental results show that the success rate of G-WALK in tracking individual users is just over 43% with the time traceability metric and around 30% with the distance traceability metric, which is obviously lower than that of L-WALK. This fact can be explained based on the probability assignment done by the common sniffing stations and speed-matching heuristics. Both these heuristics are static and memoryless, i.e., they do not consider the probability of reaching a particular state (from the initial state) in order to estimate transition probabilities from that state. As a result, the transition probabilities assigned by these heuristics would suggest a good ordering rather than an accurate overall probability estimate, which explains why L-WALK performs better as it attempts to determine the most likely order of states by making locally optimal decisions. Figure 3 shows a (partial) snapshot of the single-user tracking algorithm execution on the data collected from our experiments. In this figure, the blue color shows the groundtruth,

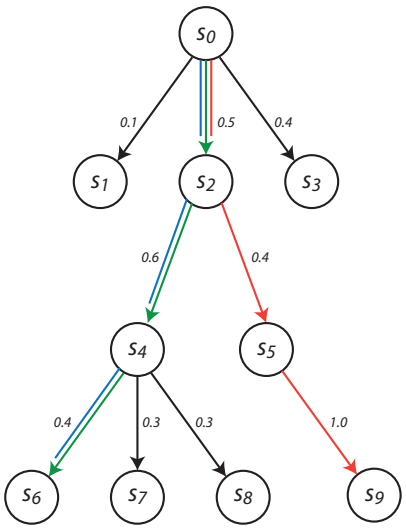

Figure 3. L-WALK **versus** G-WALK**.**

the green color indicates the path chosen by the L-WALK algorithm and the red color indicates the path chosen by the G-WALK algorithm. We can clearly see that G-WALK attempts to choose a globally optimal path, which is not the correct one, and thus results in a lower traceability. Moreover, a manual examination of a few tens of sample paths also reveals that the percentage of situations, where the overall most likely path in the graph was truly the correct path, was less than 20%, thus explaining the discrepancy between L-WALK and G-WALK. Further improvements of the likelihood heuristics and G-WALK are left as future work. L-WALK could be further improved by using reference points in order to guide the state search. Only one extra reference point suffices to increase L-WALK's average traceability by 20%. With two more reference points, the traceability nears 95%.

Fig. 2(b) presents multiple user time- and distance-traceability results. Both L-WALK and G-WALK methods yield approximately the same results: over 80% time traceability and over 60% distance traceability. The main advantage of multiple user tracking over single user tracking is that the former leads to no collisions: as all users are being tracked simultaneously, no two users can have the same next state. Another advantage is that multiple user tracking algorithms can handle the users' leaving the experiment area, and their coming back, because they have a global picture of all users. These advantages lead to a higher tracking success.

Table 2 summarizes all tracking results for the previously defined metrics. In particular, we observe that the uncertainty (u_{avg}) depends both on the data and the heuristic used and that the μ-metrics (traceability-uncertainty combined) roughly follow the τ-metrics (traceability only). Finally, the c-metric, which is defined only in the multiple user scenario, indicates a clustering error of about 8 meters. Although this is not enough to precisely track a single user, in most cases it is enough to identify the room in which the user is.

5.3. Adversary Strength

Hereafter, we evaluate the success of the proposed tracking algorithms under varying adversarial strengths. We vary adversarial strengths by systematically reducing the adversary's sniffing capabilities (see Table 3) at each step of the evaluation. At each step, we reduce the number of sniffing stations (specifically, by six additional stations), and thus the corresponding sniffed data, available for tracking. These sniffing stations are selected based on the coverage provided by each of them such that the overal coverage after their removal is maximized.

Fig. 2(c) shows the success rate of single user tracking under varying adversarial strengths. We can observe that the traceability success increases with the number of adversarial sniffing stations, which is intuitive. Interestingly, the traceability success stabilizes after a given point (31 sniffing stations), where the adversary's coverage can be considered as optimal and additional stations are not very useful.

Similarly, Fig. 2(d) shows the success rate of multiple user tracking for varying values of adversary strength. The traceability success trends for the multiple user track-

Table 2. Complete tracking results using the time, distance and clustering privacy metrics.

Users	Method	Heuristic	τ_t	τ_d	u_{avg}	μ_t	μ_d	c_{avg}
Single User	L-WALK	common stations	62%	47%	0.69	40%	37%	-
		speed matching	61%	51%	1.21	59%	41%	-
	G-WALK	common stations	46%	31%	0.33	23%	17%	-
		speed matching	43%	28%	0.80	36%	25%	-
Multiple Users	L-WALK	common stations	84%	65%	0.47	-	-	8.32
		speed matching	88%	70%	0.63	-	-	7.86
	G-WALK	common stations	83%	67%	0.52	-	-	8.78
		speed matching	81%	64%	0.56	-	-	8.10

ing case are very similar to those in single user tracking, except that in multiple user tracking, the success rate is around 20% higher compared to single user tracking over all the adversary strengths. In fact, by processing multiple users at the same time, we observe that the algorithms perform better even with significant inaccuracies in the user positions.

Table 3. Adversary strengths.

# stations	Density [$/1000m^2$]	Strength
37	1.9	100%
31	1.6	84%
25	1.3	68%
19	1.0	51%
13	0.7	35%
7	0.4	19%

There might be other adversary models for tracking that are significantly weaker than (even the passive version of) the omni-potent Dolev-Yao model but relevant from the practical point of view. To be more widely useful, such an adversary model should, of course, be applicable to many different kinds of scenarios. Finding such a model is an interesting research challenge.

5.4. Impact on Network Efficiency

While providing privacy protection, one major concern for pseudonym change mechanisms is the effect it can have on the network performance. We analyze the effect of our pseudonym change mechanism on the network performance by computing the average time spent by devices in a mix-zone and the resulting packet loss rate. We estimate the average time of a device in a mix-zone by computing the ratio of the average mixing time to the average time between two mixings for that device. We estimate the packet loss between two mix attempts by computing the ratio of the average number of packets that would be sent during a mix attempt to the average number of packets sent between two mix attempts. In our experiments, we observe that between two mixing attempts, devices spend an average of 1.5% of their network time in mix-zones, resulting in a packet loss rate of approximately 2.4%. These results show that PCA with the current parameters does not have a major effect on the network performance. However, these values do not consider the costs involved in re-establishing connections or the effect on transport protocols with congestion control mechanisms.

5.5. Traceability in Large User Clusters

Intuitively, when users organize themselves in large clusters (each cluster consisting of many mix-zones close to each other), user density within individual mix-zones also increases, thus leading to larger anonymity sets. Technically, this should result in better protection against trace reconstruction or tracking attacks. To verify this intuition, we measure the traceability of users on some selected days when the user-density is comparatively higher, for example, around the time when participants have lectures. Specifically, we define two tracking intervals, each of approximately 10 minutes, one at the beginning and the other at the end of the monitored lecture.

We plot (Fig. 4) the average time traceability, as defined in Section 5.1, and the *picked-up* ratio during these two tracking intervals. The picked-up ratio is the ratio of the number of users for which the subtraces in the first and the second tracking interval match the ground truth to the total number of users in both the intervals. The horizontal axis in the plot denotes the number of users, during a particular lecture, that can be followed by our multiple target tracking algorithm. This number is also indicative of the size of the user cluster on that particular day, although it does not directly represent the size of each mix-zone within that cluster. We plot tracking results for 9 different lectures of the same course. We can see from the results that an increase in the number of users present in the cluster decreases both the traceability and the picked-up ratio. In other words, large clusters enable users to hide from the adversary. Note that these results include users (not necessarily in equal num-

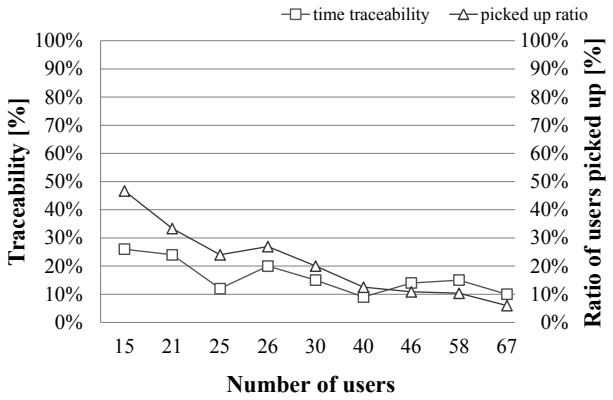

Figure 4. Traceability and picked-up ratio versus the number of users in large clusters.

bers) from all the three parameter values outlined in Table 1 and that non-deterministic factors, such as user movements, speed etc., can have an influence on the traceability, as visible from the slight irregularities in the plots.

5.6. PCA Improvements

The experimental results with the current set of PCA parameters (Table 1) show that even mobile users deploying identifier-changes are highly traceable and are prone to location privacy-related attacks. The question remains: Is it possible to choose better values (from the privacy perspective) for PCA parameters or to improve the PCA in general? In this section, we attempt to answer these questions by selecting two new sets of PCA parameters, which we consider to be significantly more aggressive with respect to privacy preservation. These parameters are outlined in Table 4. We performed simulations to verify the effectiveness of these aggressive parameter values in preventing user traceability.

Table 4. Aggressive PCA parameters.

Parameter	Aggressive	V. aggressive
Mean t_f	1200 sec	600 sec
Mean t_c	120 sec	60 sec
$Q = Q_c + Q_f$	200	500
change_time_threshold	300 sec	120 sec
neighbor_threshold	3	3

We simulated PCA with these new parameters, while retaining all original user communication and mobility patterns. These simulations build a new set of traces from the original traces and from the groundtruth (obtained during the real experiments) by correctly simulating coordinated identifier-changes or mixing on the devices at the appropriate times and locations.

Fig. 5(a) and 5(b) show the traceability results of the `L-WALK` and `G-WALK` algorithms on the simulation-based traces. These plots show that, although aggressive parameters result in better location privacy for users in general, multiple-user tracking still performs sufficiently well and is a concern to location privacy. Furthermore, both the new sets of parameters result in a significant network performance degradation, which makes it harder to implement them in practice. Network performance will only worsen for more aggressive parameter values.

Overall, the traceability results indicate that the current PCA specification, regardless of the chosen parameter values, is not very successful in preventing tracking attacks against users in mobile and pervasive networking systems. As a result, we propose three improvements to our original PCA specification, as outlined below, and evaluate each of them in a simulation setting.

1. *PCA with radio silence randomized over a larger time interval*: each device implements a radio silence delay (t_{silent}) that is randomly selected between 0 and 30 seconds, as opposed to the original specification, where it was randomly selected between 10 and 20 seconds.

2. *PCA with longer radio silence*: radio silence (t_{silent}) is observed for a longer time period (30-90 seconds).

3. *PCA with radio silence until movement detected*: radio silence is observed until the user has traveled a distance of at least d (randomly selected in the interval $[0, 20]$) meters since the mixing decision.

As before, we simulate these PCA improvements over the original experimental traces and the groundtruth obtained from the real experiments. Tracking is performed on the resulting simulated traces and the traceability results are outlined in Fig. 5(c) and 5(d). We can see that all the PCA improvements discussed above are successful against the speed matching heuristic. However, solely randomizing the radio silence, as proposed in the first two cases, is not enough to defeat the common sniffing stations heuristic due to the lower mobility and speed of users. Determining the length of the radio silence period by making use of the user's mobility (third improvement) provides protection against this heuristic. In this case, radio silence periods can sometimes be extremely long, e.g., 30 minutes or more, which could result in poor network performance and QoS.

6. Conclusion

In this paper, we have evaluated the effectiveness of mix-zone-based identifier-change mechanisms in upcoming

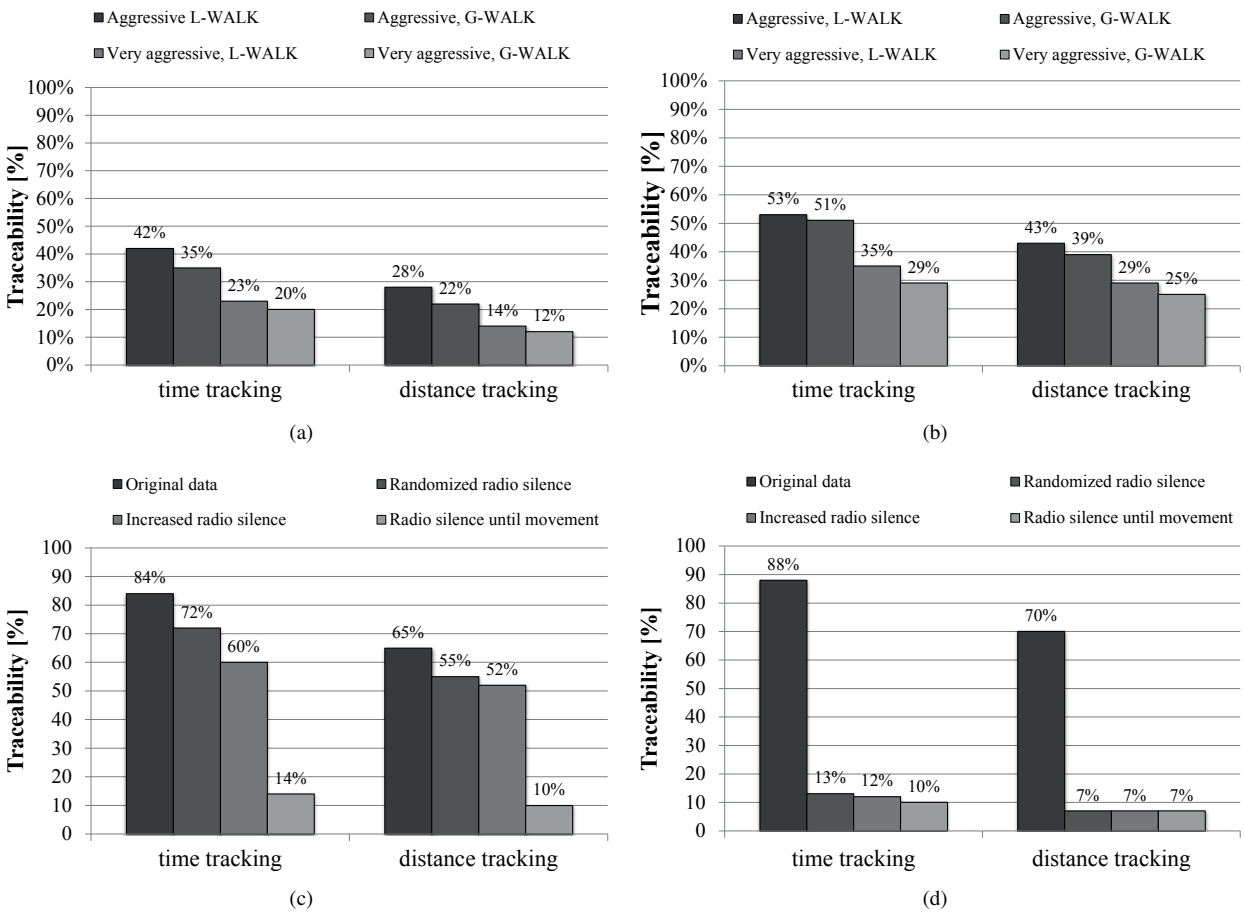

Figure 5. Results for aggressive parameters: 5(a) single user; 5(b) multiple users. Results with modified PCA in multiple user tracking: 5(c) common sniffing stations; 5(d) speed matching.

wireless and mobile systems by means of a real deployment of 80 Nokia N900 smartphones on the EPFL campus. The local passive adversary, in our case, is comprised of a wireless mesh network of 37 access points covering a $200mx100m$ rectangular area of the campus. Our adversary's goal was to reconstruct an accurate ordering of identifiers that belong to the same device in order to learn the path taken by users. For this purpose, the adversary constructed a probabilisitc tracking (Markov) model by employing intuitive heuristics on the observations from the adversarial access points. Two strategies were used to determine the most likely path taken by the users, (1) a local greedy strategy (`L-WALK`) and (2) a globally optimal strategy (`G-WALK`).

Our results have shown that, in real settings, even simple tracking strategies achieve high traceability success. By changing identifiers in an aggressive fashion, we have observed that the tracking success of the adversary reduces considerably, albeit with a significant decrease in the network performance. We have observed that a decrease in the number of adversary sniffing stations results in lower traceability. There is still a need to find a generic adversary model that is weaker than the standard Dolev-Yao model but stronger than our localized and stationary eavesdropper. Finally, we have also shown that randomizing silence periods within a mix-zone, based on time-of-change or distance travelled post-change, can vastly imrove the effectiveness of mixing in real systems.

Acknowledgments

We would like to thank Adel Aziz, Julien Herzen and Patrick Thiran for their support and guidance during the setup of the adversarial wireless mesh network. Special thanks to Andreea Anghel for implementing the PCA and to all the EPFL students who participated in the trial. Finally, we are grateful to Nokia Research Center for funding this project.

References

[1] https://foursquare.com/.

[2] http://en.wikipedia.org/wiki/Lovegetty.

[3] http://en.wikipedia.org/wiki/Bluedating.

[4] https://facebook.com/.

[5] http://pleaserobme.com/.

[6] I. Aad, M. Jadliwala, I. Bilogrevic, V. Niemi, L. Bindschaedler, J.-P. Hubaux, P. Ginzboorg, and K. Leppänen. Nokia Instant Community at EPFL: a Real-world Large-scale Wireless Peer-to-Peer Trial. Technical Report EPFL-REPORT-170421, EPFL, Lausanne, Switzerland, 2011.

[7] A. Ahtiainen, K. Kalliojarvi, M. Kasslin, K. Leppanen, A. Richter, P. Ruuska, and C. Wijting. Awareness networking in wireless environments: Means of exchanging information. *IEEE Vehicular Technology Magazine*, September 2009.

[8] T. Alpcan and S. Buchegger. Security games for vehicular networks. *IEEE Transactions on Mobile Computing*, 2011.

[9] A. R. Beresford and F. Stajano. Location privacy in pervasive computing. *Pervasive Computing*, 2003.

[10] A. R. Beresford and F. Stajano. Mix zones: User privacy in location-aware services. In *PerCom Workshop*, 2004.

[11] L. Bindschaedler. Track me if you can. Master's thesis, School of Computer and Communication Sciences, EPFL, 2011.

[12] L. Buttyán, T. Holczer, and I. Vajda. On the effectiveness of changing pseudonyms to provide location privacy in VANETs. In *ESAS*, 2007.

[13] L. Buttyán, T. Holczer, A. Weimerskirch, and W. Whyte. Slow: A practical pseudonym changing scheme for location privacy in VANETs. In *IEEE VNC*, 2009.

[14] D. Chaum. Untraceable electronic mail, return addresses and digital pseudonyms. *Comm. ACM*, 1981.

[15] D. Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *J. Cryptology*, 1988.

[16] Z. Chen, H. Shen, Q. Xu, and X. Zhou. Instant Advertising in Mobile Peer-to-Peer Networks. In *ICDE*, 2009.

[17] M. Corson, R. Laroia, J. Li, V. Park, T. Richardson, and G. Tsirtsis. Toward Proximity-aware Internetworking. *Wireless Communications*, 2010.

[18] C. Diaz, S. Seys, J. Claessens, and B. Preneel. Towards measuring anonymity. In *PETS*, 2002.

[19] D. Dolev and A. C. Yao. On the security of public key protocols. *IEEE trans. on Information Theory*, 1983.

[20] B. M. East. Iranian opposition activists hanged for protest footage. *http://www.bbc.co.uk/news/world-middle-east-12272067*, 2011.

[21] J. Eberspächer, H.-J. Vögel, C. Bettstetter, and C. Hartmann. *GSM - Architecture, Protocols and Services*. Wiley, 2008.

[22] M. Follman. "Bluetoothing" Iran's revolution. *Markfollman.com*, 2010.

[23] J. Freudiger, M. Jadliwala, J.-P. Hubaux, V. Niemi, P. Ginzboorg, and I. Aad. Privacy of community pseudonyms in wireless peer-to-peer networks. Technical Report EPFL-REPORT-170392, EPFL, Lausanne, Switzerland, 2011.

[24] J. Freudiger, M. Raya, M. Felegyhazi, P. Papadimitratos, and J.-P. Hubaux. Mix zones for location privacy in vehicular networks. In *Win-ITS*, 2007.

[25] J. Freudiger, R. Shokri, and J.-P. Hubaux. On the optimal placement of mix zones. In *PETS*, 2009.

[26] J. Freudiger, R. Shokri, and J.-P. Hubaux. Evaluating the privacy risk of location-based services. In *Financial Cryptography*, 2011.

[27] S. Gaonkar, J. Li, R. R. Choudhury, L. P. Cox, and A. Schmidt. Micro-blog: sharing and querying content through mobile phones and social participation. In *MobiSys*, 2008.

[28] M. Gerlach and F. Guttler. Privacy in VANETs using changing pseudonyms - ideal and real. In *IEEE VTC-Spring*, 2007.

[29] P. Golle and K. Partridge. On the anonymity of home/work location pairs. In *Pervasive*, 2009.

[30] M. Hachman. Peep Proposes Wireless P2P System. *PC Magazine*, 2011.

[31] B. Hoh and M. Gruteser. Protecting location privacy through path confusion. In *SecureComm*, 2005.

[32] B. Hoh, M. Gruteser, H. Xiong, and A. Alrabady. Preserving privacy in GPS traces via uncertainty-aware path cloaking. In *ACM CCS*, 2007.

[33] J. Hromkovič. *Algorithms for Hard Problems*. Springer-Verlag, 2004.

[34] L. Huang, K. Matsuura, H. Yamane, and K. Sezaki. Enhancing wireless location privacy using silent period. In *IEEE WCNC*, 2005.

[35] M. Humbert, M. H. Manshaei, J. Freudiger, and J.-P. Hubaux. Tracking games in mobile networks. In *GameSec*, 2010.

[36] M. Jadliwala, I. Bilogrevic, and J.-P. Hubaux. Optimizing mixing in pervasive networks: A graph-theoretic perspective. In *ESORICS*, 2011.

[37] M. Jadliwala, S. Zhong, S. Upadhyaya, C. Qiao, and J.-P. Hubaux. Secure distance-based localization in the presence of cheating beacon nodes. *IEEE Transactions on Mobile Computing*, 2010.

[38] C. Karlof, N. Sastry, and D. Wagner. Tinysec: a link layer security architecture for wireless sensor networks. In *SenSys*, 2004.

[39] M. Khiabani. Metro-sexual. http://bit.ly/theranMetroSexual, 2009.

[40] N. Kiukkonen, J. Blom, O. Dousse, D. Gatica-Perez, and J. Laurila. Towards rich mobile phone datasets: Lausanne data collection campaign. In *ICPS*, 2010.

[41] M. Li, K. Sampigethaya, L. Huang, and R. Poovendran. Swing & swap: user-centric approaches towards maximizing location privacy. In *WPES*, 2006.

[42] B. Palanisamy and L. Liu. Mobimix: Protecting location privacy with mix zones over road networks. In *ICDE*, 2011.

[43] Rhiain. Nokia instant community gets you social. http://conversations.nokia.com/ 2010/05/25/nokia-instant-community-gets-you-social/.

[44] R. Shokri, G. Theodorakopoulos, G. Danezis, J.-P. Hubaux, and J.-Y. Le Boudec. Quantifying location privacy: The case of sporadic location exposure. In *PETS*, 2011.

[45] R. Shokri, G. Theodorakopoulos, J.-Y. Le Boudec, and J.-P. Hubaux. Quantifying location privacy. In *IEEE Security & Privacy Symposium*, 2011.

[46] M. Stiemerling and S. Kiesel. A system for peer-to-peer video streaming in resource constrained mobile environments. In *U-NET*, 2009.

[47] Theodore S. Rappaport. *Wireless Communications: Principles and Practice, 2nd Edition*, chapter Mobile Radio Propagation: Large-Scale Path Loss. Pearson Education, Inc., 2003.

[48] B. Wiedersheim, Z. Ma, F. Kargl, and P. Papadimitratos. Privacy in Inter-Vehicular Networks: Why simple pseudonym change is not enough. In *IEEE/IFIP WONS*, 2010.

[49] H. Yoon, J. Kim, F. Tan, and R. Hsieh. On-demand video streaming in mobile opportunistic networks. In *PERCOM*, 2008.

[50] X. Zhang, H. Heys, and C. Li. An analysis of link layer encryption schemes in wireless sensor networks. In *ICC*, 2010.

Appendix

Table 5. Symbol Table.

Symbol	Definition
m	User's actual message
$t(m)$	Device time when the message m is sent
$p(m)$	Device position when message m is sent
$\pi(m)$	Device identifier when message m is sent
c	Message content
M	Set of all actual messages (sent by all users)
Q_f	Quota for forced identifier changes
t_f	Forced timer
Q_c	Quota for context-based identifier changes
t_c	Context check timer
t_{silent}	Radio silence period
Q	Total identifier-change quota
$\hat{m}$	Copy of the m observed by the adversary
$\hat{o}(\hat{m})$	Adversary AP observing $\hat{m}$
$\hat{s}(\hat{m})$	Received signal strength of $\hat{m}$
r	Actual sequence of pseudonyms
$\tilde{e}$	Event constructed from observed messages
$\hat{T}$	Set of time-stamps in the event
$\hat{O}$	Set of observing APs in the event
$\tilde{r}$	Reconstructed sequence of pseudonyms
$\tilde{m}$	Message reconstructed from an event $\tilde{e}$
S	State space
P	State transition function
$v(s)$	Speed associated with the state s
τ	Traceability metric
u	Uncertainty metric
μ	Traceability-uncertainty metric
C	Clustering metric