Post-hoc User Traceability Analysis in Electronic Toll Pricing Systems

Xihui Chen[1]*, David Fonkwe[1], and Jun Pang[2]

[1] Interdisciplinary Centre for Security, Reliability and Trust, University of Luxembourg
[2] Faculty of Sciences, Technology and Communication, University of Luxembourg

Abstract. Electronic Toll Pricing (ETP), a location-based vehicular service, allows users to pay tolls without stopping or even slowing down their cars. User location records are collected so as to calculate their payments. However, users have privacy concerns as locations are considered as private information. In this paper, we focus on user traceability in ETP systems where anonymous location records are stored by the service providers. Based on user toll payment information, we propose a post-hoc analysis of user traceability, which aims at computing a user's all possible traces. Moreover, we propose several methods to improve the effectiveness of the analysis by combining other contextual information and propose a number of optimisations to improve its efficiency as well. We develop a prototype and evaluate the effectiveness of the analysis by conducting extensive experiments on a number of simulated datasets.

1 Introduction

Electronic Toll Pricing (ETP) systems, by collecting tolls electronically, aim to reduce users' delay at toll gates and thus to increase the throughput of public transportation networks. Cars are automatically identified when passing check points. The locations of the check points and the passing time are collected and stored by toll servers in the form of location records, which are used to calculate users' payments afterwards. Nowadays, the free access to civilian Global Navigation Satellite Systems (GNSS) has upgraded ETP into a more sophisticated service. Compared to location sensing techniques, e.g., number plate recognisers, GNSS positioning covers a much wider area. This leads to smart pricing and "Pay-As-You-Drive" (PAYD), binding a user's insurance to the roads he has actually travelled [1]. Meanwhile, this also causes more privacy concerns as more user records are collected by the system. For instance, by processing location records, the toll server can learn users' home addresses or medical information [2]. User mobility pattens can also be extracted [3], which are useful to construct user profiles.

In the last few years, protecting location privacy in ETP and PAYD systems has been widely studied [1, 4–9]. The general idea is to anonymise users' records and hide the links among them. In other words, ETP systems do not leak either the owner of a location record (*unlinkability*) or the fact that two records belong to the same user (*untraceability*). In general, privacy-preserving ETP systems can be divided into two categories based on whether users' locations are stored on user devices or the toll server. The first

* This author was supported by the FNR Luxembourg under project SECLOC 794361.

type of ETP systems, e.g., see [6], where users do not send locations to toll servers, offer better privacy protection. Whereas, the introduction of sophisticated cryptographic techniques such as zero-knowledge proofs usually imposes heavy computation overheads on user devices. By contrast, the second type of ETP systems (e.g., see [5, 7]), where location records are anonymised, have less computation overheads on user devices and can also support other applications, e.g., traffic monitoring and control. However, as such ETP systems focus on protecting a user's privacy and enforcing correct toll calculation, we notice that they usually tend to ignore the threats after user payment information has been calculated or collected by the server. Once users' payment information becomes part of the adversary's knowledge, the claimed user location privacy can be reduced or even nullified. For example, if in a given user's collected location records, there is only one combination of locations which have the same cost as his tolls, then the user's privacy is completely broken. In this paper, we focus on a user's untraceability, which is considered as a stronger requirement than unlinkability, in this big type of ETP systems. We describe a user's traceability by a set of possible traces he might have travelled, and compute or *de-anonymise* such sets in a certain period (called a *toll session*) based on users' payments and other available contextual information.

Related work. De-anonymisation has been applied in many areas, e.g., datasets [10] and social networks [11, 12]. The idea of de-anonymisation is to find the correlation of two elements by utilising background information. Narayanan and Shmatikov show that visitors of a website can be identified by combining their group membership information in social networks [11]. They also demonstrate that information from different data sources can be combined to de-anonymise a user in large sparse datasets [10].

With respect to location privacy, Gruteser et al. explore the technique of multi-target tracking to link location records of a user [13, 14]. After learning one of a user's locations, the user's whole trace can thus be obtained with a high probability. However, in the context of ETP, to the best of our knowledge, this paper is the *first attempt* to investigate threats on privacy by de-anonymising users' traces based on toll payments.

Our contributions. We study the threats on user traceability in the ETP systems collecting anonymised location records. A *post-hoc user traceability analysis* using user toll payment information and such anonymised location records is proposed from the view of the adversary. We show that its effectiveness can be improved by taking into account additional information which is easily obtained, e.g., maps. For the sake of efficiency, we propose several optimisations to make the analysis more practical. We develop a prototype and evaluate the analysis on a number of simulated datasets.

Structure of the paper. After introducing our framework, the adversary model and the notion of traceability sets (Sect. 2), we first propose an algorithm calculating users' traceability sets based on the fact that a user's trace in a toll session should cost the same as his toll payment (Sect. 3). Subsequently, we explore two types of contextual information that can help the adversary to reduce the size of the traceability set. One is about temporal and spatial constraints between location records (i.e., *reachability* and *connectivity*) while the other concerns users' repetitive behaviour patterns (i.e., *session similarity*) (Sect. 4). To improve the efficiency of our algorithm, we propose three optimisations in Sect. 5. The experimental results show that our analysis is effective (Sect. 6). We conclude the paper in Sect. 7 with some research directions for the future.

2 Preliminaries

2.1 Formal Framework

Let $\mathcal{U}$ be the set of users who register an ETP service and $\mathcal{L}$ be the set of all locations that can be passed by users. In practice, time is always discrete with minutes or seconds as the minimum unit. We use $\mathcal{T}$ to denote the totally ordered set of all discrete time points whose granularities are determined by applications.

Using a location, in the scenario of ETP, we can identify the corresponding road segment between two intersections. We call this segment a *link* and denote it by a pair $\langle \ell_1, \ell_2 \rangle$, where ℓ_1 and ℓ_2 are the two end locations. When two links shared one end location, we say they are *consecutive*. Formally, let $\ell k_1 = \langle \ell_1, \ell_2 \rangle$ and $\ell k_2 = \langle \ell_1', \ell_2' \rangle$ be two links. If $\ell_2 = \ell_1'$ or $\ell_2' = \ell_1$, then they are consecutive, denoted as $\ell k_1 \sim \ell k_2$. Given a map, we use $\mathcal{LK} \subseteq \mathcal{L} \times \mathcal{L}$ to denote the set of all links on it.

A typical anonymised location record collected by toll servers can be abstracted in the form of $\langle \ell, t \rangle$, indicating that a user passed location ℓ at time t. As a user's location records transmitted on a link can be easily linked to each other, we consider links as the smallest units in our further analysis. Thus, instead of location records, we have link records, e.g., $\langle \ell k, t_e, t_x \rangle$ where t_e denotes the time the user entered ℓk and t_x is the time the user exited ℓk. To calculate users' tolls, toll servers first assign fees to link records based on a charging policy. The charging policy can be modelled as a function $f : \mathcal{LK} \times \mathcal{T} \to \mathbb{R}$, calculating the fee of a link according to the time a user entered the link, e.g., t_e.

Tracking technologies proposed in the last few years (e.g., see [14, 15]) allow us to make a further abstraction. As the adversary is able to link a user's link records together with a relatively high confidence when the user does not stop his car, these link records construct a *trip*. A *trip* is a sequence of link records a user continuously transmitted from one place to another. The order between links is determined by their time stamps. Suppose *tr* is a trip, denoted by $(\langle \ell k_1, t_{e_1}, t_{x_1} \rangle, \ldots, \langle \ell k_n, t_{e_n}, t_{x_n} \rangle)$ and t_δ is the minimum stay time of users before travelling on the next link. Then we have $\ell k_i \sim \ell k_{i+1} \wedge 0 < (t_{e_{i+1}} - t_{x_i}) \leq t_\delta$ for $1 \leq i < n$. We use $tr.startLink$ and $tr.startTime$ to denote its starting link, i.e., ℓk_1 and the corresponding entering time t_{e_1} of a trip tr. Similarly, we have $tr.endLink = \ell k_n$ and $tr.endTime = t_{x_n}$. The length of tr, the number of links in the trip, is denoted as $len(tr)$. Using link fees, we can calculate the corresponding fee for a given trip tr, i.e., $fee(tr) = \sum_{0 < i \leq len(tr)} f(\ell k_i, t_{e_i})$. For two trips tr_1 and tr_2 with $tr_1.startTime < tr_2.startTime$, the distance between them, i.e., $d(tr_1, tr_2)$ is computed as the length of the shortest path connecting $tr_1.endLink$ and $tr_2.startLink$. In the following discussion, we fix a toll session and use $\mathcal{TR}$ to represent the set of trips transmitted by all users.

During a toll session, a user's real trace can thus be represented as a set of his trips stored in $\mathcal{TR}$. We call this set a *trace*. Suppose Tr_u be the real trace of user u in a toll session. The amount of tolls that u has to pay is $cost_u$ which equals to $\sum_{tr \in Tr_u} fee(tr)$.

Example 1. Fig. 1 shows a fraction of a map. There are five locations $\ell_1, \ldots, \ell_5$, which are the positions of five intersections. We also have four links $\ell k_1, \ldots, \ell k_4$ where $\ell k_i = \langle \ell_i, \ell_{i+1} \rangle$ ($i = 1, \ldots, 4$). Suppose a user moves from ℓ_1 to ℓ_5 with one-hour stay at ℓ_3

and t_{ℓ_i} is the time the user passes ℓ_i. Then he has two trips with t_δ being 30 minutes – $(\langle \ell k_1, t_{\ell_1}, t_{\ell_2}\rangle, \langle \ell k_2, t_{\ell_2}, t_{\ell_3}\rangle)$ and $(\langle \ell k_3, t_{\ell_3}, t_{\ell_4}\rangle, \langle \ell k_4, t_{\ell_4}, t_{\ell_5}\rangle)$.

The notations used in this paper are summarised in Tab. 1

$\mathcal{U}$	set of users
$\mathcal{L}$	set of locations
$\mathcal{T}$	set of time points
$\mathcal{LK}$	set of links
$\mathcal{TR}$	set of trips
$fee(tr)$	trip tr fee
$cost_u$	user u's toll payment
Tr_u	a trace of user u
$d(tr_1, tr_2)$	distance between trips tr_1 and tr_2

Table 1: Notations.

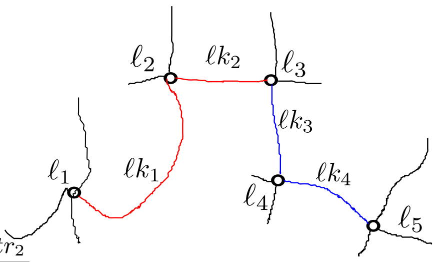

Fig. 1: An example of a user's trips.

2.2 Adversary Model

The adversary has the motivation to obtain users' travel history (i.e., traces), because extracting traces is an essential preliminary step towards further data inference, e.g., trajectory pattern mining [3], location-based recommendation, car pooling and friend finder [16, 17]. In this paper, we assume that toll servers are malicious and collude with the adversary, which makes users' location records and toll payment information part of the adversary's knowledge. This assumption is realistic as the servers may sell their databases to, e.g., advertising companies, to have additional revenues. In the sequel, we focus on *post-hoc analysis*, meaning that the adversary analyses users' traces after users' toll payment information has been calculated and agreed by the users.

Besides, we assume the adversary has access to some common knowledge such as maps, the maximal speeds of cars, etc. Such information is easy to obtain, and the adversary can extract further information from it. For example, using maps, for any two links, the adversary can learn their distance. Such information is useful for the adversary to reduce their uncertainty on users' traces.

2.3 Traceability Sets

In the last few years, a number of measurements for location privacy have been proposed (e.g., see [14, 18]). As the adversary's aim is to trace users, inspired by the notion of anonymity sets by Chaum [19], we propose *traceability sets* to measure users' traceability. A traceability set consists of all traces that are possibly linked to a given user from the adversary's point of view, including the user's real trace. We use $AS_{TR}(u)$ to denote user u's traceability set.

Without considering extra information such as maps, i.e., the adversary only has access to users' anonymous trips $\mathcal{TR}$ and users' toll payments, the initial traceability set for a user u in the given toll session can be computed as follows:

$$AS_{TR}(u) = \{Tr \subseteq \mathcal{TR} \mid \sum_{tr \in Tr} fee(tr) = cost_u\},$$

based on the fact that a user's trace should cost the same as his toll payment.

3 An Algorithm for Computing Traceability Sets

In this section, we propose an algorithm to construct users' initial traceability sets, which is inspired by a solution to the *subset sum problem* (SSP). The subset sum problem can be formulated as follows [20]: *Given a finite set $\mathcal{A}$, a weight function, $w : \mathcal{A} \to \mathbb{N}$, and a constant $s \in \mathbb{N}$, determine whether or not there exists a subset $B \subseteq \mathcal{A}$ such that $\sum_{a \in B} w(a) = s$.* This 'yes/no' decision problem has been proved to be NP-hard [21] and has no polynomial time solution so far. However, to the SSP with certain restrictions, polynomial time solutions have been proposed in the last few decades [22]. An optimal solution is proposed by Pisinger [23], which has linear computation time when each element in $\mathcal{A}$ is positive and bounded with the same constant. Let N be the size of $\mathcal{A}$ and W be the upper bound. The elements in $\mathcal{A}$ are ordered and then processed one after another. For each value in $[s-W, s+W]$, the algorithm maintains the corresponding *dominating* subset the summation of whose elements is equal to the value. The algorithm returns 'yes' if s has such a subset. In this way, the size of the intermediate set remains $2W$ and the computation complexity is thus $\mathcal{O}(NW)$.

As we have mentioned before, a user's initial traceability set is composed of the traces whose cost equals the user's toll payment and the adversary's aim is to further reduce the size of the set. Thus to compute a user's traceability set is essential for the adversary. This computation can be reduced to the SSP, thus it is also NP-hard and the corresponding computation time is exponential in users' payments. However, inspired by Pisinger's algorithm, we design an algorithm which has a good efficiency. The trips are non-increasingly ordered according to their fees and processed one after another. The algorithm maintains an intermediate set to store the subsets of earlier trips which are possible to construct a trace using the further trips to be processed. Similar to Pisinger's algorithm, the size of this intermediate set determines the number of operations needed. Therefore, we should keep it as small as possible to accelerate the computation of traceability sets.

More specifically, we take the following heuristics: (1) if the cost of the trips in a subset is larger than the user's payment, then the subset is removed as all fees are positive and the cost will never be reduced to the user's payment; (2) if the future trip with the minimum fee is added into a subset and the total cost of the resulted subset is larger than the payment, the subset can also be removed; (3) if all future trips are added into a subset and the new subset's cost is still less than the payment, then it is also removed. Alg. 1 shows the algorithm in more details.

The set U is the intermediate set which consists of all plausible subsets up to the current trip and initially it only consists of the empty set. For any trip in $\mathcal{TR}$, before adding to a subset $S \in U$, the algorithm checks the plausibility of U first with heuristics (2) and (3) (line 12). Note that $restFee$ is the cost of the future trips that have not been processed and $minFee$ is the minimum fee among all trips. After the trip is added, if the cost of the new subset equals $cost_u$, the set is added to the result $AS_{TR}(u)$. It is added into the set T when the corresponding cost is smaller than $cost_u$ and removed, otherwise. At the end of each loop, U is updated by unionising T and $restFee$ is subtracted by the trip's fee. After obtaining $AS_{TR}(u)$, the adversary starts to reduce it by using additional contextual knowledge, which is discussed in the following section.

Algorithm 1 An algorithm to build an initial traceability set for a user u.

```
1: FUNCTION: buildTraceSet
2: INPUT: $\mathcal{TR}$, $cost_u$
3: OUTPUT: $AS_{TR}(u)$;
4: $sort(\mathcal{TR}$, non-increasing);
5: $minFee = \min_{tr \in \mathcal{TR}} fee(tr)$;
6: $restFee = \sum_{tr \in \mathcal{TR}} fee(tr)$;
7: $AS_{TR}(u) = \emptyset$;
8: $U \leftarrow \{\emptyset\}$;
9: for all $tr \in \mathcal{TR}$ do
10:   $T = \emptyset$;
11:   for all $S \in U$ do
12:     if $restFee + \sum_{tr' \in S} fee(tr') < cost_u \vee minFee + \sum_{tr' \in S} fee(tr') > cost_u$ then
13:       $U = U/\{S\}$;
14:     else
15:       if $fee(tr) + \sum_{tr' \in S} fee(tr') = cost_u$ then
16:         $AS_{TR}(u) = AS_{TR}(u) \cup \{S \cup \{tr\}\}$;
17:       else
18:         if $fee(tr) + \sum_{tr' \in S} fee(tr') < cost_u$ then
19:           $T = T \cup \{S \cup \{tr\}\}$;
20:         end if
21:       end if
22:     end if
23:   end for
24:   $U = U \cup T$; $restFee = restFee - fee(tr)$;
25: end for
26: return $AS_{TR}(u)$;
```

4 Reducing Traceability Sets

Once the adversary explores more information, he can improve the post-hoc analysis of a user's traceability by reducing the traceability sets.

4.1 Reachability and Connectivity

A user's real trace in a toll session has certain constraints among trips, which can be used to remove traces in his traceability set. We discuss two of such constraints – *reachability* and *connectivity*.

Both of the constraints benefit from maps, which are easy to obtain, especially after free high-precision maps such as Google maps become accessible to ordinary users. Given two positions, the adversary can compute the distance between them. Combining with users' maximum speed, we have the first constraint – reachability between trips. Intuitively, given two trips, if along the moving direction, users cannot move to the starting point of the later trip from the ending point of the earlier trip even with his maximum speed, then the later trip is considered not reachable from the earlier one. We use $maxSpeed_u$ to denote the maximum speed allowed by user u's vehicle. Recall that $d(tr_1, tr_2)$ is the distance between two trips.

Definition 1 (Reachability). *Let tr_1 and tr_2 be two trips in $\mathcal{TR}$ and $tr_1.endTime < tr_2.startTime$. We say tr_2 is reachable for user u from tr_1 (denoted as $tr_1 \rightsquigarrow tr_2$) if and only if*

$$tr_2.startTime - tr_1.endTime > \frac{d(tr_1, tr_2)}{maxSpeed_u}.$$

This relation is reflexive, transitive but not symmetric because the connection between two trips may be unidirectional.

The second constraint is connectivity which is defined on two *successive trips*. Two trips are successive if in a trace, there are no other trips started between their starting time. For any two successive trips, they are *connected* if the earlier trip's ending point coincides with the other trip's starting point. In practice, due to the errors from positioning devices, even two positions from a static user may be different. Furthermore, it is possible that some vacant time exists between two trips before the first location record of the later trip is sent to the server. Therefore, we introduce a tolerance parameter d_δ to indicate the maximum distance allowed between two connected points.

Definition 2 (Connectivity). *Let tr_1 and tr_2 be successive trips and $tr_1.endTime < tr_2.startTime$. We say tr_1 is connected to tr_2 (denoted as $tr_1 \to tr_2$) if and only if*

$$d(tr_1, tr_2) < d_\delta.$$

In practice, we cannot expect any two successive trips are connected because the tolling road sections are not always connected. Therefore, we define a metric called *connection rate* to measure the proportion of connected trips in a user's trace. Given a trace Tr, its connection rate $cr(Tr)$ can be calculated as follows:

$$cr(Tr) = \frac{|\{\langle tr_i, tr_j\rangle \,|\, tr_i \in Tr, tr_j \in Tr \text{ s.t. } tr_i \to tr_j\}|}{|Tr|-1}.$$

Let c be the pre-defined minimal connection rate. Making use of the above two constraints, i.e., trip reachability and connectivity, the adversary can further reduce a user u's traceability set $AS_{TR}(u)$ to

$$\{Tr \subseteq \mathcal{TR} \,|\, (\forall_{\{tr_i, tr_j\} \subseteq Tr} tr_i \rightsquigarrow tr_j \vee tr_j \rightsquigarrow tr_j) \wedge cr(Tr) > c \wedge \sum_{tr \in Tr} fee(tr) = cost_u\}.$$

4.2 Session-to-Session Similarity

Users tend to have regular mobility patterns, which has been greatly discussed in the literature (e.g., see [2, 3]). This implies that a user's traces in different sessions should also have some similarity. Our aim is to show how the adversary can explore the similarity between toll sessions to reduce his uncertainty on users' traces. As users' daily activities are fixed to some extent, the places they linger are also fixed and in fact contained among the starting links and the ending links of trips. Since users' real stay places occur repetitively from session to session, this makes them have higher occurrence frequencies in sessions compared to other places. Our idea is to rank traces in a traceability set based on links' *similarity degrees* determined by their occurrence in sessions.

Given a link ℓk, we say that ℓk has appeared in a trace Tr denoted as $\ell k \underline{\in} Tr$ if there exists a trip $tr \in Tr$ such that $tr.startLink = \ell k \vee tr.endLink = \ell k$.

Definition 3 (A Link's Similarity Degree). *Let $AS_{TR1}, \ldots, AS_{TRn}$ be a user u's traceability sets in n sessions and ℓk be a link appearing in at least one trace in any traceability set. The* similarity degree *for the link ℓk (i.e., $sim(\ell k)$) is measured as:*

$$sim(\ell k) = \mid \{AS_{TRj} \mid \exists Tr \in AS_{TRj} \text{ s.t. } \ell k \sqsubseteq Tr\} \mid.$$

It is easy to see that $1 \leq sim(\ell k) \leq n$. For a traceability set, we can evaluate the similarity degree of each trace in it by the average similarity degree of the links appearing in the trace as follows:

$$\mathcal{S}(Tr) = \frac{\sum_{\ell k \sqsubseteq Tr} sim(\ell k)}{\mid Tr \mid}.$$

For the traces in a traceability set, the adversary can thus have an approximate distribution over them based on their similarity degrees. This distribution represents the probabilities of the user to travel on each trace in the set. Given a user u and AS_{TRi}, we use $p_u(T_i = Tr)$ to denote the probability of trace Tr being u's real trace in session i, which is computed as follows:

$$p_u(T_i = Tr) = \frac{\mathcal{S}(Tr)}{\sum_{Tr' \in AS_{TRi}} \mathcal{S}(Tr')}.$$

This distribution enables the adversary to have a better guess on the user' real trace.

5 Improving Efficiency of the Post-hoc Analysis

From the above discussion, we can see that the initial traceability set is essential for the subsequent analysis. Although in Alg. 1, we have used the methodology of dynamic programming to improve its efficiency, Alg. 1 is still time-consuming as the computation complexity remains exponential in users' payments. Recall that we can keep the intermediate set U in Alg. 1 as small as possible to reduce the calculation time. Along this direction, we propose three optimisations – *on_the_fly trip non-overlapping detection* (OTF), *weak_user_first* (WUF) and *parallel_traversing* (PTR).

OTF. This optimisation explores a relation between trips – *non-overlapping*. Intuitively, as users never travel at two trips at the same time, in a trace, any two trips should be generated one after another and their travel periods should not overlap.

Definition 4 (Non-overlapping). *Let tr_1 and tr_2 be two trips in $\mathcal{TR}$. Trip tr_1 and tr_2 do not overlap (denoted as $tr_1 * tr_2$) if and only if:*

$$tr_1.endTime \leq tr_2.startTime \ \vee \ tr_2.endTime \leq tr_1.startTime.$$

This relation is irreflexive and symmetric, but not transitive.

From the definitions of reachability and non-overlapping, we can see for any two trips tr_1 and tr_2, if $tr_1 \rightsquigarrow tr_2$ then $tr_1 * tr_2$ as the time interval between two reachable trips can never be negative. However, compared with reachability check, non-overlapping detection is more efficient as only the simple comparison between time stamps is involved. Whereas, the distance between two trips has to be calculated to verify reachability, which is time-consuming especially in the case when the number of trips is large. Therefore, due to its efficiency, we add non-overlapping as another condition when deciding whether to add a new subset into the intermediate set or the traceability set (line 15 and 18).

WUF. As mentioned before, the time to compute a user's traceability set will become unacceptable when his payment is big.[3] However, if users with small fees (called *weak users*) are processed first, then users with large payments can be processed based on the trip set with weak users' traces removed. A smaller input $\mathcal{TR}$ in Alg. 1 thus results in less computation time and less memory consumption. Moreover, if we calculate a user's traceability set starting from the weakest user, a set of partitions of the trip set is obtained, each of whose blocks corresponds to a user's trace. A user's traceability set can then be constructed by all his traces occurring in those partitions. Alg. 2 gives more details of this calculation.

Algorithm 2 Weak_user_first Algorithm.

```
1: PROCEDURE WUF
2: INITIAL: $\forall u \in \mathcal{U}, AS_{TR}(u) = \emptyset$;
3: INPUT: $\mathcal{TR}$, $P = (u_1, \ldots, u_n)$
4:
5: $AS = \mathsf{buildTraceSet}(\mathcal{TR}, cost_{u_1})$;
6: for all $Tr \in AS$ do
7:     $\forall i > 1, AS(i)_{old} = AS_{TR}(u_i)$
8:     WUF($\mathcal{TR}/Tr, (u_2, \ldots, u_n)$);
9:     if $\exists i \leq n, AS(i)_{old} \neq AS_{TR}(u_i)$ then
10:        $AS_{TR}(u_1) = AS_{TR}(u_1) \cup \{Tr\}$;
11:    end if
12: end for
13: return;
```

Users are first ordered in a sequence P according to their payments where users with smaller payments are put in front of the sequence. The algorithm takes $\mathcal{TR}$ and sequence P as input and its termination implies the accomplishment of the calculation of all users' traceability sets. Users' traceability sets are initially set empty. The weakest user u_1's traceability set AS is first calculated based on our algorithm discussed in Sect. 3 (line 5). Then for each trace $Tr \in AS$, we update the rest of the users' traceability sets by recursively calling the algorithm but with the set of trips not contained in Tr. The increment of one of the other users' traceability sets indicates that Tr is a trace that can be included in a partition of $\mathcal{TR}$. It is also added into $AS_{TR}(u_1)$ (line 7-11). When all traces in AS of the weakest user have been tested, all users' traceability sets are computed and the whole algorithm terminates.

PTR. In Alg. 1, for each trip in $\mathcal{TR}$, the set U is traversed sequentially from the beginning to the end (line 11 in Alg. 1) and updated at the end of the loop. So the computation time grows linearly with the size of U, which increases exponentially in the number of trips processed. However, we find the order of the subsets in U has no impact on the

[3] In some extreme cases, the algorithm may run out of the memory.

update of U at line 24. This observation allows us to process multiple subsets in U at the same time and leads to a parallel implementation.

Let $\{c_1, \ldots, c_k\}$ be the set of available processors with shared memory, where k is the number of processors. At the beginning of each loop (line 9 in Alg. 1), we split the intermediate set U evenly into sets $U_1, \ldots, U_k$. Processor c_i is then assigned with U_i, executes the operations at line 11-24 with U_i as input and returns the updated U_i as output, denoted as U'_i. Afterwards, we update U as $\bigcup_{i\in\{1,\ldots,k\}} U'_i$, which will be used in the next loop. Although ideally we can accelerate the algorithm by around k times, due to communication and task scheduling overhead we cannot achieve this in practice.

6 Experimental Results

We have conducted experiments to evaluate our analysis. First, we evaluate the optimisations and show that their combination has the best performance. Then we evaluate the methods to reduce users' traceability sets.

Setting of the experiments. The experiments are performed on synthetic datasets generated by SUMO, a tool simulating urban mobility [24]. Users travel in Luxembourg city with an area of 51.46 km^2. There exist public real trajectory databases such as the one in [25, 26], but we cannot perform comprehensive analysis, e.g., similarity and connectivity due to the lack of user profiles and the incompleteness of location records. Furthermore, as our purpose is to test the effectiveness of the proposed post-hoc analysis, synthetic mobility databases are sufficient.

For each user, we automatically generate a profile including his places of interest and whether he has kids. Users' daily activities are generated based on such profiles by adding some randomness. SUMO takes such activity sequences as input and produces users' real traces according to factors such as real-time traffic and vehicle types. In general, the datasets used in our experiments consist of the trips of 50 users for 10 days, with each day as a toll session. On average, a user's real trace on a specific day has about five trips. The minimum stay time t_δ is set to 30 minutes. We implement a prototype in Python and run the experiments on a computation cluster of nodes with 12 2.26GHz Intel Xeon Cores, sharing 24GB memory.

Efficiency of the optimisations. We start with evaluating the optimisations. Fig. 2 shows the computation time used to construct users' traceability sets based on datasets belonging to different number of users. We have four settings in terms of whether optimisations OTF, WUF and PTR are implemented. If no optimisations are implemented, in about 12 hours, the algorithm can only handle 10 users while using the optimisation OTF the algorithm can compute the traceability sets for 14 users. The improvement by WUF is more obvious. A dataset with 16 users is analysed. In our prototype, we simplify PTR by splitting the intermediate set (i.e., U in Alg. 1) only when its size first reaches 60,000 instead of in every loop due to the restriction of Python. This does not give us the best efficiency but still saves more than half of the computation time further. In the following experiments, we use the algorithm with all optimisations implemented.

Effectiveness of the reducing methods. We proceed to show the performance of the reducing methods as discussed in Sect. 4. We start with reachability and connectivity.

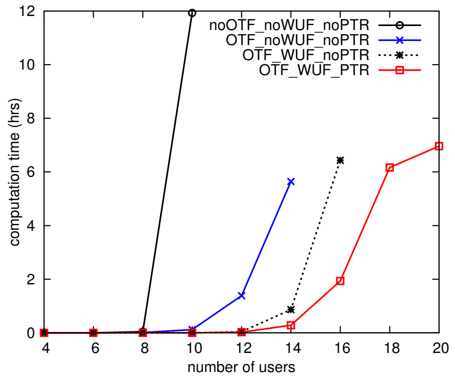

Fig. 2: Evaluation of the optimisations.

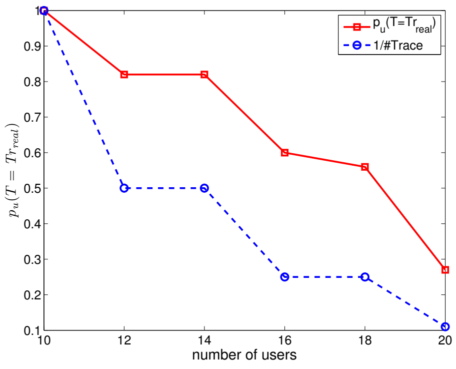

Fig. 3: Session-to-session similarity.

Both of these two methods need to calculate the distance between successive trips. However, reachability is verified for each pair of successive trips while connectivity check relies on connection rates that can only be obtained after all pairs have been processed. Thus, we can perform reachability check first to reduce total check time. This is because the traces violating reachability can be immediately removed and there is no need to check connectivity on them any more. As in our datasets, users are always simulated to start the next connected trip from the ending link of the previous trip, thus the parameter d_δ is set to zero.

Tab. 2 summarises the results of running our algorithm on datasets with increasing sizes. Reachability check and connectivity check with connection rates 0.3 and 0.7 are executed sequentially. We use *reduction rate* to measure the number of traces removed from initial traceability sets by each check in addition to the previous one. We use '#init-Trace' to denote the number of traces in the initial traceability sets (without performing any checks), and '#Trace' and '#Reduc' to represent the number of traces left and the reduction rate after each check. Specifically, using reachability check, the proportion of removed traces stays around 15% when the number of users is larger than 14. Connectivity check removes most of the redundant traces. When the connection rate is 0.3, an average of 65% of the traces are removed while the connection rate of 0.7 will further reduce 13.5% of the traces on average. Although a large proportion of traces can be removed, we can see that more traces are calculated when the datasets have more users. This also leads to more remaining traces. Thus, we can conclude that when more users are involved in an ETP system, it provides better privacy protection to users.

To show the effectiveness of session similarity, we fix a user and put him in groups with increasing sizes. We compare the size of his traceability set in a session. The similarity degree is calculated based on ten toll sessions. Let Tr_{real} be the user's real trace in the given session. Tab. 3 shows the size of the user's traceability set and the corresponding posterior probabilities of his real trace. Fig. 3 depicts the changes of the real trace's probability before and after session similarity is explored by the adversary. Session similarity does reduce the adversary's uncertainty about the real trace. For instance, the real trace has a probability 60% in 4 traces which is much larger than the

Table 2: Effectiveness of reachability and connectivity analysis.

#user	#initTrace	Reachability		Connectivity 0.3		Connectivity 0.7	
		#Trace	%Reduc	#Trace	%Reduc	#Trace	%Reduc
4	17	17	0%	8	53%	4	24%
6	48	48	0%	16	67%	5	23%
8	99	97	2%	25	73%	7	18%
10	1501	1387	8%	180	80%	14	11%
12	9801	9043	8%	943	83%	49	9%
14	12757	11726	8%	1175	83%	53	9%
16	234167	195750	16%	13706	78%	246	6%
18	1158788	975397	16%	61753	79%	712	5%
20	3800390	3276691	14%	189302	81%	3378	5%
22	6085206	5269807	13%	282897	82%	4268	5%

uniform probability 25% (see the column where the number of users is 16). It is also clear that the datasets with more users provide better privacy protection as the size of the traceability set grows from 1 to 9 when the group size increases from 10 to 20.

Table 3: Effectiveness of toll session similarity.

	#User=10	User=12	#User=14	#User=16	#User=18	#User=20
#Trace	1	2	2	4	4	9
$p_u(T = Tr_{real})$	100%	82%	82%	60%	56%	27%

7 Conclusion & Future Work

In this paper, we presented a post-hoc analysis of users' traceability based on toll payment information within ETP systems with central toll servers collecting anonymous travel records. As far as we know, this has not been addressed in the literature.

We first proposed an algorithm based on Pisinger's solution to the subset sum problem to compute traceability sets. Then we presented methods to reduce the sizes of traceability sets using reachability and connectivity of trips. Subsequently, we specified how to utilise users' mobility pattern among toll sessions, i.e., session-to-session similarity, to reduce the adversary's uncertainty about users' real traces. To improve the efficiency, we have developed three optimisations. We have also implemented a prototype to evaluate the effectiveness of our analysis on simulated user traces. The experimental results have shown that the post-hoc analysis is effective to trace users.

There are still several ways to improve our work. First, the design of mobility datasets has limitations. For instance, users have a fixed set of places of interest in different sessions. This simplifies our calculation especially when checking connectivity. The influence of similarity between user profiles is another future work. The results can help a user to choose groups which offer him better privacy protection. Other information can be used to further improve the effectiveness of our analysis as well, e.g., the frequency of paths taken to a public place. So far, the prototype is only implemented to show how effective the analysis is. We plan to further improve its efficiency.

References

1. Troncoso, C., Danezis, G., Kosta, E., Preneel, B.: PriPAYD: privacy friendly pay-as-you-drive insurance. In: Proc. ACM Workshop on Privacy in the Electronic Society (WPES), ACM (2007) 99–107
2. Ma, Z., Kargl, F., Weber, M.: Measuring long-term location privacy in vehicular communication systems. Computer Communications **33**(12) (2010) 1414–1427
3. Giannotti, F., Nanni, M., Pinelli, F., Pedreschi, D.: Trajectory pattern mining. In: Proc. 13th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD), ACM (2007) 330–339
4. de Jonge, W., Jacobs, B.: Privacy-friendly electronic traffic pricing via commits. In: Proc. 5th Workshop on Formal Aspects in Security and Trust (FAST). Volume 5491 of LNCS., Springer (2008) 143–161
5. Popa, R.A., Balakrishnan, H., Blumberg, A.J.: VPriv: Protecting privacy in location-based vehicular services. In: Proc. 18th USENIX Security Symposium, USENIX Association (2009) 335–350
6. Balasch, J., Rial, A., Troncoso, C., Geuens, C.: PrETP: Privacy-preserving electronic toll pricing. In: Proc. 19th USENIX Security Symposium, USENIX Association (2010) 63–78
7. Chen, X., Lenzini, G., Mauw, S., Pang, J.: A group signature based electronic toll pricing system. In: Proc. 7th Conference on Availability, Reliability and Security (ARES), IEEE CS (2012) to appear
8. Garcia, F., Verheul, E., Jacobs, B.: Cell-based roadpricing. In: Proc. 8th European Workshop on Public Key Infrastructures, Services and Applications (EuroPKI). Volume 7163 of LNCS., Springer (2011) 106–122
9. Meiklejohn, S., Mowery, K., Checkoway, S., Shacham, H.: The phantom tollbooth: Privacy-preserving electronic toll collection in the presence of driver collusion. In: Proc. 20th USENIX Security Symposium, USENIX Association (2011)
10. Narayanan, A., Shmatikov, V.: Robust de-anonymization of large sparse datasets. In: Proc. 29th IEEE Symposium on Security and Privacy (S&P), IEEE CS (2008) 111–125
11. Narayanan, A., Shmatikov, V.: De-anonymizing social networks. In: Proc. 30th IEEE Symposium on Security and Privacy (S&P), IEEE CS (2009) 173–187
12. Wondracek, G., Holz, T., Kirda, E., Kruegel, C.: A practical attack to de-anonymize social network users. In: Proc. 31st IEEE Symposium on Security and Privacy (S&P), IEEE CS (2010) 223–238
13. Gruteser, M., Hoh, B.: On the anonymity of periodic location samples. In: Proc. 2nd Conference Security in Pervasive Computing (SPC). (2005) 179–192
14. Hoh, B., Gruteser, M., Xiong, H., Alrabady, A.: Preserving privacy in GPS traces via uncertainty-aware path cloaking. In: Proc. 14th ACM Conference on Computer and Communications Security (CCS), ACM (2007) 161–171
15. Shokri, R., Theodorakopoulos, G., Boudec, J.Y.L., Hubaux, J.P.: Quantifying location privacy. In: Proc. 32nd IEEE Symposium on Security and Privacy (S&P), IEEE CS (2011) 247–262
16. Trasarti, R., Pinelli, F., Nanni, M., Giannotti, F.: Mining mobility user profiles for car pooling. In: Proc. 17th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD), ACM (2011) 1190–1198
17. Zheng, Y., Zhang, L., Ma, Z., Xie, X., Ma, W.: Recommending friends and locations based on individual location history. ACM Transactions on the Web **5**(1) (2011) 5
18. Chen, X., Pang, J.: Measuring query privacy in location-based services. In: Proc. 2nd ACM Conference on Data and Application Security and Privacy (CODASPY), ACM (2012) 49–60

19. Chaum, D.: The dining cryptographers problem: Unconditional sender and recipient untraceability. Journal of Cryptology **1**(1) (1988) 65–75
20. Pérez-Jiménez, M.J., Riscos-Núñez, A.: Solving the subset-sum problem by P systems with active membranes. New Generation Computing **23**(4) (2005) 339–356
21. Garey, M., Johnson, D.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman & Co. (1990)
22. Oltean, M., Muntean, O.: Solving the subset-sum problem with a light-based device. Natural Computing **8**(2) (2009) 321–331
23. Pisinger, D.: Linear time algorithms for knapsack problems with bounded weights. Journal of Algorithms **33**(1) (1999) 1–14
24. Behrisch, M., Bieker, L., Erdmann, J., Krajzewicz, D.: Sumo - simulation of urban mobility: An overview. In: Proc. 3rd Conference on Advances in System Simulation (SIMUL). (2011) 63–68
25. Xiao, X., Zheng, Y., Luo, Q., Xie, X.: Finding similar users using category-based location history. In: Proc. 18th ACM SIGSPATIAL Symposium on Advances in Geographic Information Systems (GIS), ACM (2010) 442–445
26. Yuan, J., Zheng, Y., Zhang, C., Xie, W., Xie, X., Sun, G., Huang, Y.: T-drive: Driving directions based on taxi trajectories. In: Proc. 18th ACM SIGSPATIAL Symposium on Advances in Geographic Information Systems (GIS), ACM (2010) 99–108