Design and Formal Analysis of A Group Signature Based Electronic Toll Pricing System *

Xihui Chen[1]†, Gabriele Lenzini[1], Sjouke Mauw[1,2] and Jun Pang[2]

[1] Interdisciplinary Centre for Security Reliability and Trust,
University of Luxembourg, Luxembourg
xihui.chen, gabriele.lenzini@uni.lu
[2] Faculty of Science, Technology and Communication,
University of Luxembourg, Luxembourg
sjouke.mauw,jun.pang@uni.lu

Abstract

Location-based vehicle services have been enduring a rapid growth with the prevalence of GNSS technologies, nowadays freely available for everyone. Given the nature of location data, privacy is of prime importance in services such as electronic tolling pricing (ETP) and pay-as-you-drive. In this paper, we first propose a new electronic toll pricing system based on group signatures – GroupETP which achieves a good balance between privacy and overhead imposed upon user devices. Second, we give a comprehensive formal analysis of GroupETP. Our analysis, conducted by ProVerif, requires some abstraction of the system and extensions to the existing analysis methods. The results show that GroupETP satisfies the desired properties – correctness, accountability and unlinkability.

1 Introduction

Electronic Toll Pricing (ETP) systems, by collecting tolls electronically, aim to eliminate delays due to queuing on toll roads and thus to increase the throughput of transportation networks. For instance, the E-ZPass system deployed in the US is estimated to reduce the total delay at toll plazas by 85 percent [2]. By exploiting the availability of free Global Navigation Satellite Systems (GNSS), traditional ETP is evolving into a more sophisticated location-based vehicular service. Users' trajectories are collected in the form of GNSS positions and the passing time, and subsequently used to calculate the tolls of each user. In this way, it can offer smart pricing, e.g., by charging less who drive on uncongested roads or during off-peak hours. Insurance companies can also bind insurance premiums to roads that their users actually use, and offer a service known as "Pay-As-You-Drive" (PAYD) [3]. As locations are considered as sensitive and private information, ETP and PAYD systems raise users' privacy concerns. It is out of users' control that how their locations will be used. For example, locations gathered by E-Zpass have already been used in courts and applications to extract traffic statistics. The research such as user profiling based on spatio-temporal datasets [4, 5, 6] aggravates the situation. One of the steps of user profiling is to extract users' places of interest and mobility patterns, which users obviously want to keep secret.

In the last few years, secure ETP and PAYD systems have been widely studied [3, 7, 8, 9, 10, 11]. They can roughly be divided into two categories based on whether locations are stored in user devices or collected by central toll servers. PriPAYD [3], PrETP [9], the cell-based solution described in [10], and Milo [11] belong to the first category. In these systems, locations and tolls are managed by user devices while servers are allowed to process only aggregated data. In the second category we find VPriv [8],

*This paper is an extended version of the work originally presented at the 6th International Conference on Availability, Reliability and Security (ARES'12), Prague, Chzech Republic, August 2012 [1].

†Supported by the National Research Fund, Luxembourg (SECLOC 794361)

where the server stores a database of users' travel history, and the ETP system described in [7], where the server collects the hash values of trip records.

Hiding locations from servers drastically reduces the concerns over location privacy. However, it also imposes an overhead on user devices as they have to store travel records and construct proofs that they have not cheated, e.g., by zero-knowledge proofs. The availability of location databases collected by servers, e.g., in VPriv, can improve applications such as traffic monitoring and control, although the integration of multiple systems should be carried out carefully. Whereas, more effort is required to preserve users' location privacy than those without explicit locations stored.

We propose a new ETP system called GroupETP [1], which is based on our evaluation of the strengths of the existing systems. It follows the design principles of VPriv [8] in order to have a good balance between location privacy and overhead on user devices. Users are divided into groups and use group signature schemes to sign their locations. If users and the server cannot reach an agreement on the final amount of tolls, an authority is then involved to open the group signatures and solve the dispute. As only the authority has the right to open the location signatures, users' anonymity is protected in groups against the server. With the division of groups and one-round calculation of tolls, GroupETP reduces the amount of exchanged information as well as the computation overhead compared to VPriv.

As the design of security systems is error-prone, we believe that the security claims of an ETP system should be formally verified before deployment in practice. In this paper, we model GroupETP with the applied pi calculus, and perform a formal analysis of the system by leveraging the previous analysis of vehicular services, e.g., mix-zones [12] and VPriv [13]. With the aid of the protocol analysis tool ProVerif [14], we show that GroupETP satisfies all the desired security properties – *correctness*, *accountability* and *unlinkability*.

Structure of the paper. Sect. 2 briefly describes our ETP system GroupETP [1]. In Sect. 3, we introduce shortly the applied pi calculus which are used to model the system in Sect. 4. In Sect. 5 we show the methods to analyse the system with ProVerif as well as the analysis results. We conclude our paper in Sect. 6 with some discussions and ideas for future work.

2 The GroupETP System

2.1 System Model

Principals. The system consists of four principals: *users*, *their cars* with on board units (*OBUs*), *the authority*, and *the toll server*.

Users own and drive cars, and are also responsible for toll payments. To be entitled to use the electronic tolling service, a user brings his car to the authority, which registers the car and installs an OBU on it. The authority is a governmental department trusted by both users and the toll server. It also builds up the group signature scheme and manages groups of users. An OBU computes locations using GNSS, e.g., GPS (Global Positioning System), and stores them, for example, in a USB stick or a TPM (Trusted Platform Module), which interfaces the OBU and contains security information. It transmits location data to the toll server, which is a logical organisation that can be run by multiple agents in practice. The server collects location data and computes the fee of each location record. It can also contact the authority to resolve disputes when some users cheat on their tolls.

Adversary model. With regards to the deployment environment of ETP systems, the possible threats can come from: 1) manipulated OBUs, which generate false location records; 2) dishonest users, who (partially) avoid paying their road usage; 3) dishonest toll servers, who intend to increase their revenue and breach users' privacy; 4) the honest but curious authority.

Assumptions. Considering the practice of ETP systems, we make an assumption that users and the server have motivation to obtain economic benefits during toll payment and toll collection (*Assumption 1*). Users tend to pay fewer tolls than those they should pay according to their road usage and the server wants no economic loss. For instance, dishonest servers can perform any actions to satisfy their strong economic motivation and curiosity. They can deviate from the protocols and collude with other attackers. The attackers considered in the system follow the Dolev-Yao intruder model [15] (*Assumption 2*). Specifically, they have full control over the network, which means they can eavesdrop, block and inject messages anywhere at anytime. However, an encrypted message can never be opened unless they have the right key. We assume that location tuples are transmitted to toll servers anonymously (*Assumption 3*). This can be achieved by the architecture in [16], for instance, which uses a communication service provider to separate authentication from data collection.[1] The authority is supposed to be curious but not to collude with any other participants, meaning that it tries to peek users' privacy only based on the data learnt during the execution of the protocols it is involved in (*Assumption 4*).

It has been shown that users' moving traces can be reconstructed from anonymised positions, e.g., by multi-target tracking techniques [17] or taking into account users' mobility profiles [18], and users' private information can thus be inferred [19]. However, we observed from the experiments in the literature that tracking users remains difficult in practice, especially when the intervals between transmissions are big (about one minute) and the number of travelling users is not small. Therefore, similar to VPriv, in this paper, we focus on privacy leakage from ETP systems without considering the above mentioned trace reconstruction techniques (*Assumption 5*).

2.2 Security Properties

In addition to reducing communication and computation overheads, GroupETP deploys proper measures to protect honest users and servers. Referring to what other ETP systems achieve (e.g., [3, 8, 9]), we give the following security properties which GroupETP should satisfy:

Correctness. Users pay their road usage correctly and the server collects the right amount of tolls.

Accountability. If a malicious action that deviates from the specification of the system occurs, sufficient evidence can be gathered to identify its originator.

Unlinkability. The adversary cannot link a given location record to its generator.

2.3 Cryptographic Primitives

Group Signature Schemes. Group signatures [20] provide the signers anonymity among a group of users. A group signature scheme consists of group members and a group manager. The task of the group manager is to organise the group, set up the group signature infrastructure and reveal signers if needed. The signature of a message, signed by a group member, can be verified by others based on the group public key while the identity of the signer remains secret.

Group signature schemes contain at least the following five functions: SETUP, JOIN, SIGN, VERIFY and OPEN. Function SETUP initialises the group public key, the group manager's secret key and other related data. The procedure JOIN allows new members to join the group. Group members call function SIGN to generate a group signature based on their secret keys. The VERIFY function makes use of the group public key to check if a given signature is signed by a group member. Function OPEN determines the signer of a signature based on the group manager's secret key.

We take group signature schemes as an essential building block of our system because they have the following properties, which effectively meet our security goals.

[1] Thus identification based on message transmission is out of the scope of our paper.

- CORRECTNESS Signatures produced by a member using SIGN must be accepted by VERIFY.
- UNFORGEABILITY Only group members can sign messages on behalf of the group.
- ANONYMITY Given a valid signature of some message, identifying the actual signer is unfeasible for everyone but the group manager.
- UNLINKABILITY Deciding whether two different valid signatures were computed by the same group member is unfeasible.
- EXCULPABILITY Neither a group member nor the group manager can sign on behalf of other group members.
- TRACEABILITY The group manager is always able to open a valid signature and identify the actual signer.
- COALITION-RESISTANCE A colluding subset of group members cannot generate a valid signature that the group manager cannot link to one of them.

There are some other properties we desire as well, e.g., efficiency and dynamic group management. Efficiency concerns the length of signatures and computation time of each function, which determines the feasibility of our system. Dynamic group management enables users to join or quit their current groups at any time when they are new or not satisfied (e.g., see [21]).

In the last decade, efficient group signature schemes with new fancy features have been developed, e.g., group message authentication [22] and group signcryption [23]. Some of the schemes may improve the security of our system. For instance, an efficient group signcryption scheme can prevent attackers from eavesdropping users' location signatures over the network. In the description of our system, we will make use of an abstract version of group signature schemes as any group signature scheme with the required security properties can be adopted.

Public Key Cryptographic Primitives. We adopt a public key cryptosystem with classic security properties for applications of confidentiality and digital signatures. Furthermore, we assume that the public key encryption is probabilistic in order to prevent possible off-line dictionary attacks.

Cryptographic Hash Function. In the system, we also use cryptographic hash functions, which are publicly known and satisfy the minimum security requirements – preimage resistance, second preimage resistance and collision resistance.

2.4 The ETP System

Overview. Our system is organised into four phases. The first phase is about the service subscription and set-up. A user first signs a contract with a toll server in order to get access to the toll service, and establishes a security communication channel with the server. Afterwards, when users contact the authority to join a group, the authority assigns them to groups according to a group division policy (see Sect. 6). Clients' private keys for group signatures are also established during the communication with the authority. At the end of this phase, the server is informed of the groups containing its users and the corresponding group public keys.

The second phase is about collecting location data. During driving, OBUs compute their locations and time, which together with the group name form *location tuples*. OBUs periodically send location tuples and the corresponding group signatures on the hash values of the location tuples (called *location signatures*) to the server, who stores them in its location database. In order to increase entropy of the hash values, a random number is generated and added in every location tuple.

The third phase is about calculating tolls. At the end of a toll session, each user contacts the server using a user interface through browsers (not OBUs). According to the public charging policy, a user calculates the fees of his location tuples and his toll payment subsequently by adding them up. The server then collects all users' payments.

The fourth phase is about resolving a dispute. This phase takes place *only* when the sum of users' payments *in a group* is not equal to the sum of all location tuples' fees. The authority is involved to determine misbehaving participants. The server sends all location signatures and location tuples with their fees to the authority. When location signatures are opened, for each user, the authority calculates the fees belonging to his location tuples, whose sum is compared to the committed payment of the user. An inequality indicates misbehaviour from either the server or the user. The authority then contacts the user to ask for proofs. Based on the received proofs, the authority can find out who originated the mistake and decide the type of the misbehaviour. If the user has cheated, he has to pay their unpaid tolls to the server (possibly with an additional fine). If the server has misbehaved, it will be punished by the authority as well.

Protocol specifications. Here we specify the four protocols that implement the phases of our system, namely: *Set-up*, *Driving*, *Toll Calculation*, and *Dispute Solving*. In the following discussion, we fix a group $\mathcal{G}$ of users. Furthermore, we use $\{M\}_{pk_X}$ to denote the encrypted message of M with the public key of X and $[M]_{sk_X}$ to represent the signature of X on message M. A group signature of user u in $\mathcal{G}$ on M is denoted as $[M]_{gsk_u}$. From a signature, the corrrespondign message M cannot be determined and M is requried to verify the signature.

Phase 1: Set-up. This protocol accomplishes two tasks. The first is to establish the public key infrastructure between the users, the server and the authority. The second task is to set up the group infrastructure.

We make use of two secrets to achieve the security goal of this phase – *pin codes* and *serial numbers*. The former are generated by the server for users to prove their legal access to the toll service, while a serial number is issued with each OBU as a secret between the authority and a user. We take user u as an example. Let pin be his pin code and sn the serial number. During the authentication process between u and the authority, m is a fresh nonce generated by u. The Set-up protocol can be depicted as follows:

$$\begin{aligned}
u \to S: &\ \{u, pk_u, [pin]_{sk_u}\}_{pk_S} \\
S \to u: &\ \{[pk_u, u]_{sk_S}\}_{pk_u} \\
u \to A: &\ \{[pk_u, u]_{sk_S}, pk_u, u, sn, S, m\}_{pk_A} \\
A \to u: &\ \{gpk_{\mathcal{G}}, \mathcal{G}, m\}_{pk_u}
\end{aligned}$$

The user starts with sending his public key to the server and meanwhile his signature on pin is added in order to authenticate himself to the server. Upon receiving the user's public key, the server checks u's signature with the public key and the shared secrecy pin. If valid, the server replies with its signature on the key as a ticket which the user can use to prove the agreement of the server on his public/private key pair. The user sends the next message to the authority as a request to join a group composed of customers subscribing the service of the server S. The secrecy sn allows the authority to authenticate the user and the nonce m enables the user to authenticate the authority and ensure that the group public key in the reply uniquely corresponds to his request. Upon receiving the group public key, the user continues with the JOIN process to set up the private keys to generate group signatures.

We omit the last step where the server learns from the authority its users' groups and the group public keys, since such information can be made public.

Phase 2: Driving. The driving protocol specifies how users periodically transmit location tuples and location signatures to the server. Let $\langle \ell, t, r, \mathcal{G} \rangle$ be a location tuple where r is a random number and $h(M)$ be the hash value of message of M. A message from user u, a member of group $\mathcal{G}$, is denoted by $(\langle \ell, t, r, \mathcal{G} \rangle, [h(\ell, t, r)]_{gsk_u}$.

$$u \to S : \langle \ell, t, r, \mathcal{G} \rangle, [h(\ell, t, r)]_{gsk_u}.$$

After receiving this message, the server verifies $[h(\ell, t, r)]_{gsk_u}$ using the group public key $gpk_{\mathcal{G}}$. If valid, the received message is stored. The hash function and random numbers added are used to keep location tuples secret from the curious authority (see Phase 4).

Phase 3: Toll Calculation. This protocol aims to reach an agreement on toll payments between the server and its users. Let $\mathcal{R}_u$ be the locations which u has travelled and are stored on the USB stick. The user starts with calculating his toll payment $cost_u$ in toll session sid. For each $\langle \ell, t, r \rangle$ in $\mathcal{R}_u$, the user computes its fee $f(\ell, t)$ according to the server's public charging policy modelled by function f and obtains $cost_u$ by adding all fees up, i.e., $cost_u = \sum_{\langle \ell,t,r \rangle \in \mathcal{R}_u} f(\ell, t)$. Then the user sends to the server his signature on $cost_u$ and session identifier sid, which indicates that user u's toll payment in toll session sid is $cost_u$. After receiving u's message, the server verifies the signature before sending back its signature on u's toll payment. In addition to prove the acceptance of the user's payment, the server's signature also works as proof of the user's accomplishment of the toll calculation phase. The protocol can be described as follows:

$$\begin{aligned} u \rightarrow S: &\ \{cost_u, [cost_u, sid]_{sk_u}\}_{pk_S} \\ S \rightarrow u: &\ \{[cost_u, sid, u]_{sk_S}\}_{pk_u} \end{aligned}$$

Phase 4: Dispute Resolving. In this protocol, with the help of the authority, the server finds cheating users and the amount of unpaid tolls. The server initiates dispute resolving only when, with respect to a group, the sum of committed payments is not equal to the sum of fees of all location tuples. In practice, as users tend to pay less, the server asks for dispute resolution only when it has collected less tolls. Let $\mathcal{L}$ be the set of location tuples of group $\mathcal{G}$, then the condition can be formally described as

$$\sum_{u \in \mathcal{G}} cost_u < \sum_{\langle \ell,t,r \rangle \in \mathcal{L}} f(\ell, t).$$

A dispute resolution involves the authority, who can link a location signature to its signer. At the beginning of the dispute resolution, the server constructs two sets $\mathcal{S}$ and $\mathcal{T}$. Set $\mathcal{S}$ consists of the hash values of location tuples, the corresponding fees, and the location signatures that the server has received in Phase 2:

$$\mathcal{S} = \{\langle h(\ell, t, r), f(\ell, t), [h(\ell, t, r)]_{gsk_u} \rangle \mid \forall (\ell, t, r) \in \mathcal{L}, \forall u \in \mathcal{G}.\}$$

$\mathcal{T}$ consists of the users' committed toll payments in Phase 3:

$$\mathcal{T} = \{\langle u, cost_u, [cost_u, sid]_{sk_u} \rangle \mid \forall u \in \mathcal{G}\}.$$

Subsequently, the server sends $\mathcal{S}$, $\mathcal{T}$ and its signature on $\mathcal{S}$ to the authority

$$S \rightarrow A: \ \mathcal{S}, \mathcal{T}, [\mathcal{S}, sid]_{sk_S}.$$

Upon receiving the message, the authority starts to compute users' tolls based on $\mathcal{S}$ and find the inconsistency with users' committed toll payments. This process is described as function $SvrDisRes$ shown in Alg. 1. We use $checksign(sign, m, pk)$ to check if the $sign$ is a signature of m using pk and group signature functions VERIFY and OPEN work as described in Sect. 2.3. The check on set $\mathcal{T}$ (line 4-7) and verification of the signature on $\mathcal{S}$ (line 8-9) and location signatures (line 11-12) exclude the possibility of modifying users' toll payments by the malicious server. Each user's toll payment in $\mathcal{S}$ (i.e., $toll_u$) is computed in lines 16-17. When it is not larger than the user's committed one (i.e., $cost_u$), the user has paid the amount of tolls that the server asks for. In other words, the user's committed cost has covered all his location tuples in $\mathcal{S}$. Otherwise, the server or the user (or both) is cheating (line 19). For instance, the server may increase the fees of some location tuples or add fake location tuples in $\mathcal{S}$, while some

Algorithm 1 Function $SvrDisRes$

1: **Input**: $\mathcal{S}, \mathcal{T}, signS$
2: **Output**: res
3: $res := \emptyset$; $\mathcal{S}_u = \emptyset$; $toll_u := 0$;
4: **for all** $(u, cost, sign) \in \mathcal{T}$ **do**
5: **if** $checksign(sign, (cost, sid), pk_u) = false$ **then**
6: **return** 'check of $\mathcal{T}$ failed' ;
7: **end if**
8: **end for**
9: **if** $checksign(signS, (\mathcal{S}, sid), pk(S)) = false$ **then**
10: **return** 'check of integrity of $\mathcal{S}$ failed' ;
11: **end if**
12: **for all** $\langle hashLoc, feeLoc, gsign\rangle \in \mathcal{S}$ **do**
13: **if** VERIFY$(gsign, hashLoc) = false$ **then**
14: **return** 'Faked location signatures' ;
15: **else**
16: $u :=$OPEN$(gsign)$;
17: $toll_u := toll_u + feeLoc$;
18: $\mathcal{S}_u := \mathcal{S}_u \cup \{\langle hashLoc, feeLoc, gsign\rangle\}$;
19: **end if**
20: **end for**
21: **for all** $toll_u > cost_u$ **do**
22: $res := res \cup \{(u, toll_u - cost_u)\}$;
23: **end for**
24: **return** res

users are also possible to have committed smaller payments. For any user u with $toll_u > cost_u$, there is a pair $(u, toll_u - cost_u)$ in the result res. Moreover, all corresponding tuples of user u in $\mathcal{S}$ are stored in set $\mathcal{S}_u$.

After getting res, the authority needs to verify its correctness. In other words, the related users should try to prove their innocence. The authority sends a private message to ask each user appearing in res to initiate the dispute solving protocol with it. The server and the adversary cannot eavesdrop on this private channel. A deadline for dispute solving is also included. Any user who misses the deadline has to pay the rest of his tolls indicated by res. For the sake of simplicity, we omit the cryptographic details in the following description of the protocol:

$A \to u:$ $\mathcal{S}_u$
$u \to A:$ $\mathcal{R}'_u, [\mathcal{R}'_u, sid]_{sk_u}$ where $\mathcal{R}'_u = \{\langle \ell, t, r\rangle \mid \exists \langle h(\ell, t, r), fee, gsign\rangle \in \mathcal{S}_u \wedge f(\ell, t) < fee\}$
$A \to u:$ $[r_u, sid]_{sk_A}$ where $r_u = UsrDisRes(\mathcal{R}'_u, res, \mathcal{S}_u)$.

The user finds the set of location tuples $\mathcal{R}'_u \in \mathcal{R}_u$ with larger fees in $\mathcal{S}_u$, and sends it back to the authority. The authority then determines how many tolls the related users still need to pay by function $UsrDisRes$, which is shown in Alg. 2. First, the authority checks the integrity of $\mathcal{R}'_u$ by verifying the user's signature (line 4-5). Then for each location tuple $\langle \ell, r, t\rangle \in \mathcal{R}'_u$, the authority finds the tuple $\langle hashLoc, fee, gsign\rangle \in \mathcal{S}_u$ where $hashLoc = h(\ell, t, r)$, and accumulates the extra fee added by the server, i.e., $fee - f(\ell, t)$ (line 6-9). The result δ is the amount of tolls that the server has added to u's real tolls. By subtracting the user's committed payment $cost_u$, we obtain the rest of tolls the user still needs to pay, i.e., r_u (line 10) which is called *dispute resolution* in the following discussion.

Algorithm 2 Function $UsrDisRes$

Input: $\mathcal{R}'_u$, $signRu$, $\mathcal{S}_u$
Output: r_u
$\delta := 0$;
if $checksign(signRu, (\mathcal{R}'_u, sid), pk_u) = false$ **then**
    return 'check of $\mathcal{R}'_u$ failed';
end if
for all $\langle \ell, t, r \rangle \in \mathcal{R}'_u$ **do**
  if $\exists \langle h(\ell, r, t), fee, gsign \rangle \in \mathcal{S}_u$ **then**
      $\delta_u := \delta_u + (fee - f(\ell, t))$;
  end if
end for
$r_u := (u, toll_u - cost_u - \delta_u)$;
return r_u

For each user u, $r_u = (u, 0)$ indicates that the server has misbehaved and the user is innocent while $r_u = (u, toll_u - cost_u)$ means the server is honest and the user paid less. If $toll_u - cost_u - \delta_u < 0$, the user has paid what the server asks for (i.e., all his location tuples in $\mathcal{S}$). Otherwise, both the user and the server have misbehaved. At last, the authority constructs a set $\mathcal{R} = \{(u, v) \mid v = toll_u - cost_u - \delta_u \wedge v > 0\}$, consisting of all misbehaved users' dispute resolution and sends it back to the server. With the authority's signature on $\mathcal{R}$, the server proves the authenticity of the resolution to users, which forces them to pay their unpaid tolls. Note that after resolving, the authority only learns the location tuples with manipulated fees given by the server and the number of location records of each user in that particular group. If the misbehaved server were captured, the authority might enforce a punishment policy and make this information public. The server then has to undertake some economic loss and its reputation is thus damaged.

After executing our protocol, the server collects no less tolls than it asks for. This is why a server who wants no loss of tolls should not throw away any location tuples in $\mathcal{L}$ which is is the set of locations that the server has collected in the toll session. Otherwise, the user whose location tuple(s) is (are) omitted, may pay less by committing a cost larger than what the server asks for but smaller than what he should pay.

We refer readers to [1] for the formal definitions of the security properties and a detailed theoretical security analysis. In this paper, we resort to formal verification techniques to check the properties.

2.5 Discussion

Comparison with VPriv. As mentioned in Sect. 1, GroupETP resembles VPriv [8] in that the server collects locations. However, VPriv imposes a relatively high overhead to users and the server. Compared with VPriv, in GroupETP, the communication overhead between users and the server is reduced. GroupETP does not require the server to provide the set of location tuples with fees to users during toll calculation, which is usually large even for groups of small size. Second, we apply the principle of separation of duties, namely the authority takes the responsibility to find misbehaved users or server. Hence, the server and users are released from a heavy computational overhead by avoiding running zero-knowledge proof protocols as in VPriv. Resolving disputes needs to open all location signatures, which is time consuming for the authority. However, with punishment policies and the accountability property of our system, the authority can have a very low frequency of resolving disputes. Last but not least, we consider a malicious server with more power, which is not passive but can perform active

attacks, such as increasing fees or other attacks to learn users' whereabouts.

Group management. A good group management policy can improve the protection of users' privacy in our system. In principle, groups should be chosen to maximise the difficulty for the adversary to construct users' traces. One way to achieve this goal is to group people according to 'similarity' criteria based on multi-level hierarchical structure, as proposed in [24]. For instance, at the root level, we have the group of all users in a city. Subgroups at the next level contain those that usually travel in the same region. At lower levels the subgroups can include users having a similar driving style. Other factors can be considered as well, e.g., driving periods, car models, etc. The information needed to group people is collected by the authority at the moment of registration. The provision of such information is not compulsory but users are encouraged if they desire a better privacy protection.

Dynamic group management, which enables users to change their group memberships, is also necessary. For instance, users move to another city or they are not satisfied with the current groups. To find the optimal group size which can protect users' anonymity is our future work. Note that if a user joins multiple groups, the similarity between his travel records of the groups would decrease his anonymity.

Tamper resistant devices vs. spot checks. In order for the system to provide correct data, it must be ensured that OBUs are not manipulated by users, e.g., to transmit false locations. We consider two possible solutions to this problem. One way is to utilise devices that are tamper resistant. However, users can always turn off the device. Therefore, as discussed in VPriv and PrETP, we can use sporadic random spot checks that observe some physical locations of users. A physical observation of a spot check includes location, time and the car's plate number. Let $\langle \ell, t, pn \rangle$ be an observation of car pn whose owner is user $c \in \mathcal{G}$. Then there should be at least one location record $\langle \ell', t' \rangle$ of group $\mathcal{G}$ such that $\mid t, t' \mid < \epsilon/2$ and $\mid \ell, \ell' \mid < \gamma \cdot \mid t, t' \mid$ where ϵ is the interval between two transmissions and γ is the maximum speed of vehicles. If there are no such location tuples, then the server can determine that user c has misbehaved. Otherwise, the server could send the tuples with nearby locations to the authority to check if one of these belongs to c. According to [8], a small number of spot checks with a high penalty would suffice.

3 The Applied Pi Calculus

The applied pi calculus is a process calculus used to formally model concurrent systems and their interactions. It provides an intuitive way to model cryptographic protocols. In this section we briefly describe the syntax and semantics of the applied pi calculus. For more details, we refer the reader to [25].

The applied pi calculus assumes an infinite set of names (used to model communication channels and other atomic data), an infinite set of variables and a set of functions (used to model cryptographic primitives). *Terms* model messages which can be names, variables or functions applied to other terms. An equational theory E is used to define the equivalence between terms. A protocol is modelled as a set of processes defined as follows:

$P, Q, R ::=$	processes
0	null process
$P \mid Q$	parallel composition
$!P$	replication
$\nu\ n; P$	name restriction
if $M =_E N$ then P else Q	conditional
$\{M/x\}$	active substitution
$\text{in}(u, x); P$	message input
$\text{out}(u, M); P$	message output.

Processes consist of plain processes, which are built up similar to the pi calculus and extended processes which contains variable restrictions and active substitutions. By restricting variables, we can bound a process to a name or a variable. Active substitution $\{M/x\}$ replaces x with M. We use "let $x = M$ in P" to denote $P\{M/x\}$. A process is closed if all variables are bound either by an input or a restriction. We use $C[\_]$ to denote an evaluation context (a process with a whole) and $C[P]$ is a process obtained by filling the whole with process P. The applied pi calculus assumes the Dolev-Yao intruder [15] who controls the network.

The operational semantics of the applied pi calculus contains *structural equivalence*, *internal reduction* (denoted as $\rightarrow$) and *labelled reduction* (denoted as $\xrightarrow{\alpha}$). The structural equivalence defines the equivalence relations between two processes which differ only in structure. Internal reductions mean a process can execute an action without interaction with the context, while labelled reductions mean a process interacts with the context. Transition $A \xrightarrow{\alpha} B$ indicates that A executes α and the result is B.

There are many equivalence relations defined in the applied pi calculus. In this paper, we use *observational equivalence*, which captures the indistinguishability between two processes in any adversary contexts. To define it, we use $A \Downarrow a$ to denote that process A can send a message on a.

Definition 1 (Observational equivalence ($\approx$) [25])**.** *Observational equivalence is the largest symmetric relation $\mathcal{R}$ between closed extended processes with the same domain such that $A\mathcal{R}B$ implies:*

1. *if $A \Downarrow a$ then $B \Downarrow a$;*
2. *if $A \rightarrow^* A'$ then $B \rightarrow^* B'$ and $A' \mathcal{R} B'$ for some B';*
3. *$C[A] \mathcal{R} C[B]$ for closing evaluation context C.*

However, observational equivalence is hard to use in practice due to the quantification over contexts. Labelled bisimilarity is thus introduced, which is easier to verify either manually or automatically and coincides with observational equivalence [25].

4 Modelling

We use the applied pi calculus to model the GroupETP system. We simplify the system to have only one server and one group. Furthermore, we suppose that each user sends only one location record to the server during driving for the sake of simplicity. More importantly, if GroupETP works properly and preserves users' anonymity in groups in this extreme case, then it will also satisfy the required properties in general cases. As spot checking has been formally analysed in [13] which shows its effectiveness in preventing users' manipulation on OBUs, we assume that location records sent in the driving phase are trustworthy. In this section, we start with the description of the functions introduced and then proceed to explain the processes used in our analysis.

4.1 Signatures and equational theory

A group signature scheme consists of five functions, among which we only need to formally model three of them. The function SETUP and JOIN are captured in our modelling by assuming the availability of the group secret keys of users (gskey), the group public key (gpkey) and the group secret key of the group manager (gmskey). We introduce some functions in order to reduce the number of names in the modelling and thus keep the models simple to read. A user's group secret key is denoted by function $\mathsf{gsk}(sk)$ where sk is the user's asymmetric private key for secure communication. Because of the secrecy of sk, only the user knows his group secret key. Since for a group, the group manager has

only one group secret key corresponding to the group public key, we use $\mathsf{gpk}(y)$ to denote the group public key where y is the secret key of the group manager of the same group.

Recall that a user requires his group secret key and the group public key to sign a group signature on a string of bits. Hence, we model function SIGN by gsign with three arities whose types are gskey, gpkey and bitstring, respectively. Due to the secrecy of the user's group secret key, only the user can generate a group signature on his own behalf. The function is as follows:

$$\mathsf{fun\ gsign}(\mathsf{gskey}, \mathsf{gpkey}, \mathsf{bitstring}) : \mathsf{bitstring}.$$

The function VERIFY checks the validity of a group signature using the group public key and the string of bits signed. As long as the signature is signed on the string using the group public key, it will return true. It is captured by function checkGSign as follows:

$$forall\ x : \mathsf{gskey}, y : \mathsf{gpkey}, z : \mathsf{bitstring};\ \mathsf{checkGSign}(\mathsf{gsign}(x, y, z), y, z) = \mathsf{true}.$$

The group manager uses OPEN to learn the signer of a group signature, which requires his secret key and the group public key. As the group public key can be constructed with the manager's group secret key, we omit it in the modelling function open. We use a user's public key $\mathsf{pk}(x)$ to identify him where x is his private key for communication. Only the manager can open a group signature due to the secrecy of the manager's group secret key. The function open is as follows:

$$forall\ x : \mathsf{skey}, y : \mathsf{gmskey}, z : \mathsf{bitstring};\ \mathsf{open}(\mathsf{gsign}(\mathsf{gsk}(x), \mathsf{gpk}(y), z), y, z) = \mathsf{pk}(x).$$

Hash functions, encryptions and signed messages are modelled by functions hash, encand sign. Meanwhile decryption and verifying signatures are modelled by functions dec and checksign following the standard modelling methods.

We use $\mathsf{f}(l)$ and $\mathsf{costU}(l)$ to denote the real fee of a location record l and a smaller amount of tolls committed by a cheating user, respectively. So we have $\mathsf{costU}(l) < \mathsf{f}(l)$. Detailed equations could be found in the ProVerif code in [26].

4.2 Processes

In the following discussion, we suppose a group with m registered users. We do not give the process of the server as it is assumed to collude with the adversary in GroupETP and we can consider it as part of the adversary. Therefore, the secrets of the server are part of the adversary's knowledge.

Main process. The main process is represented in Fig. 1. This process first generates the private keys of the server, the authority and each user which are sk_S, sk_A, and sk_{u_i} ($1 \leq i \leq m$), respectively. The process then generates a location record and a boolean value for each user, e.g., l_i and $cheat_i$ ($1 \leq i \leq m$). The variable $cheat_i$ determines whether the user u_i will cheat on his payment since GroupETP assumes users to be honest except for cheating on payments. Then the manager's group secret key is generated. We public the group public key and users' public key on the public channel ch as they are part of the adversary's knowledge. The server's private key is also put on the public channel ch as it is part of the adversary. In the end, the main process launches m copies of user sub-process and one authority sub-process.

User process. The user process is described in Fig. 2. During the driving phase, the user sends his location records to the server anonymously. Due to the abstraction in the applied pi calculus that all channels are anonymous, we do not have to explicitly build such an anonymous channel [27]. Recall

$P \triangleq$ new sk_S; new sk_A; new sk_{u_1}; ...; new sk_{u_m};
new l_1; ...; new l_m; new $cheat_1$; ...; new $cheat_m$;
new $gmsk_A$;
let $gpk = \mathsf{gpk}(gmsk_A)$ in
let $pk_A = \mathsf{pk}(sk_A)$ in
let $pk_{u_1} = \mathsf{pk}(sk_{u_1})$ in ... let $pk_{u_m} = \mathsf{pk}(sk_{u_m})$ in
$\mathsf{out}(ch, gpk) \mid \mathsf{out}(ch, sk_S) \mid \mathsf{out}(ch, pk_A) \mid \mathsf{out}(ch, pk_{u_1}) \mid \ldots \mid \mathsf{out}(ch, pk_{u_m}) \mid$
$User(sk_{u_1}, pk_A, cheat_1, l_1, gpk) \mid \ldots \mid User(sk_{u_m}, pk_A, cheat_m, l_m, gpk) \mid$
$!Authority(pk_S, sk_A, pk_{u_1}, \ldots, pk_{u_m}, gmsk_A)$

Figure 1: The main process.

that $cheat$ determines the honesty of the user. In the toll calculation phase, a user sends $\mathsf{f}(l)$ to the server if $cheat$ is false. Otherwise, he sends a smaller payment $\mathsf{costU}(l)$. Afterwards, the user starts waiting for the challenge from the authority for dispute resolution which is captured by the process $disputeRes$ shown in Fig. 3. The message from the authority comes from a private channel in order to prevent the adversary from distinguishing honest and dishonest users. This channel is modelled by $\mathsf{pchA}(\mathsf{pk}(sk_u))$ where function pchA is private and out of the adversary's knowledge. If the group signature in the challenge is valid, then the user answers with his location record l. Finally he learns the resolution result from the channel ch and he continues with the payment process pay if it is valid.

$User(sk_u, pk_A, pk_S, cheat, l, gpk) \triangleq$
let $gsig_u = \mathsf{gsign}(\mathsf{gsk}(sk_u), gpk, \mathsf{h}(l))$ in $\mathsf{out}(ch, (l, gsig_u))$;
$\mathsf{in}(ch, cheat)$;
if $cheat =$ false then
let $sig_u = \mathsf{sign}(sk_u, \mathsf{f}(l))$ in
$\mathsf{out}(ch, (\mathsf{pk}(sk_u), \mathsf{f}(l), sig_u))$;
$\mathsf{in}(ch, sign_S)$;
if $\mathsf{checksign}(pk_S, (\mathsf{pk}(sk_u), \mathsf{f}(l)), sign_S) =$ true then $disputeRes(sk_u, l)$
else let $sig_u = \mathsf{sign}(sk_u, \mathsf{costU}(l))$ in
$\mathsf{out}(ch, (\mathsf{pk}(sk_u), \mathsf{costU}(l), sig_u))$;
$\mathsf{in}(ch, sign_S)$;
if $\mathsf{checksign}(pk_S, (\mathsf{pk}(sk_u), \mathsf{costU}(l)), sign_S) =$ true then $disputeRes(sk_u, l)$.

Figure 2: The user process.

$disputeRes(sk_u, l) =$ $\mathsf{in}(\mathsf{pchA}(\mathsf{pk}(sk_u)), (hash, toll, gsig))$;
if $\mathsf{checkGSign}(gsig, gpk, \mathsf{h}(l)) =$ true then
$\mathsf{out}(\mathsf{pchA}(\mathsf{pk}(sk_u)), l)$;
$\mathsf{in}(ch, (res, resSig))$;
if $\mathsf{checksign}(pk_A, (\mathsf{pk}(sk_u), res), resSig) =$ true then pay.

Figure 3: The user dispute resolution process.

Authority process. Fig. 4 shows the authority process. The authority process receives all the location records, the server's signature sig_S on them and each user' signature $sigCost_i$ on his tolls $cost_i (1 \leq i \leq m)$. In this process, the authority only receives the first m records as there are at most m valid location records from honest users and the server cannot generate any on behalf of them. Furthermore, the users' commitments are put in the ascending order of their owners' identities, i.e., $pk_{u_1}, \ldots, pk_{u_m}$. The authority checks the validity of group signatures and the cost commitments. If all are valid, then the authority inserts an entry for each user consisting of his identity pk_{u_i} and his tolls $cost_i$ in the table commitCosts. Afterwards, the authority opens each group signature and learns the originator pk_j. By reading the table, the authority obtains the corresponding user's committed payment. In the end, a challenging sub-process $commUA$ (see Fig. 5) is initiated for each user to find out the amount of tolls that the user has not paid. This sub-process corresponds to the user dispute resolution process $disputRes$ (see Fig. 3).

The $commUA$ process contacts a user u through the private channel $\mathsf{pchu}(\mathsf{pk}(sk_u))$. It first checks whether there is an agreement between the server and the user by comparing the committed number of tolls ($commitCost$) with the amount required by the server $toll$. If they are not equal, i.e., $commitCost <> toll$ ('<>' represents the inequality between terms), the hashed location record as well as the group signature are sent to the user. After receiving the user's response containing his real location rl, the process compares $toll$ and $commitCost$ with the real cost of rl and issues the rest of tolls the user has to pay according to Alg. 2.

$$
\begin{array}{l}
Authority(pk_S, sk_A, pk_{u_1}, \ldots, pk_{u_m}, gmsk_A) \triangleq \\
\quad \mathsf{in}(ch, ((hash_1, toll_1, gsig_1), \ldots, (hash_m, toll_m, gsig_m), sig_S)); \\
\quad \mathsf{in}(ch, ((cost_1, sigCost_1), \ldots, (cost_m, sigCost_m))); \\
\quad \mathsf{let}\ S = ((hash_1, toll_1, gsig_1), \ldots, (hash_m, toll_m, gsig_m))\ \mathsf{in} \\
\quad \mathsf{if}\ \mathsf{checksign}(pk_S, S, sig_S) = \mathsf{true}\ \&\& \\
\quad\quad \mathsf{checkGSign}(gsig_1, \mathsf{gpk}(gmsk_A), hash_1) = \mathsf{true}\ \&\& \\
\quad\quad\quad \ldots\ \&\&\ \mathsf{checkGSign}(gsig_m, \mathsf{gpk}(gmsk_A), hash_m) = \mathsf{true}\ \&\& \\
\quad\quad \mathsf{checksign}(pk_{u_1}, cost_1, sigCost_1) = \mathsf{true}\ \&\&\ \ldots\ \&\&\ \mathsf{checksign}(pk_{u_m}, cost_m, sigCost_m) = \mathsf{true}\ \mathsf{then} \\
\quad\quad\quad \mathsf{insert}\ \mathsf{commitCosts}(pk_{u_1}, cost_1); \\
\quad\quad\quad \ldots \\
\quad\quad\quad \mathsf{insert}\ \mathsf{commitCosts}(pk_{u_m}, cost_m); \\
\quad\quad\quad \mathsf{let}\ (pk_1, \ldots, pk_m) = (\mathsf{open}(gsig_1, gmsk_A, hash_1), \ldots, \mathsf{open}(gsig_m, gmsk_A, hash_m)\ \mathsf{in} \\
\quad\quad\quad \mathsf{get}\ \mathsf{commitCosts}(= pk_1, commitCost_1)\ \mathsf{in} \\
\quad\quad\quad \ldots \\
\quad\quad\quad \mathsf{get}\ \mathsf{commitCosts}(= pk_m, commitCost_m)\ \mathsf{in} \\
\quad\quad\quad commUA(pk_{u_1}, commitCost_1, hash_1, gsig_1, toll_1, sk_A)\ | \\
\quad\quad\quad \ldots \\
\quad\quad\quad |\ commUA(pk_{u_m}, commitCost_m, hash_m, gsig_m, toll_m, sk_A).
\end{array}
$$

Figure 4: The authority process.

5 Analysis

In this section, we give the analysis of GroupETP in terms of the security properties – correctness, accountability and unlinkability using the analysis tool ProVerif.

$commUA(pk_u, commitCost, hash, gsig, toll, sk_A) \triangleq$
  if $commitCost <> toll$ then
    $\mathsf{out}(\mathsf{pchu}(pk_u), (hash, toll, gsig))$;
    $\mathsf{in}(\mathsf{pchu}(pk_u), msgU)$;
    let $(= pk_u, rl) = msgU$ in
    if $\mathsf{h}(rl) = hash$ then
      if $toll = \mathsf{f}(rl)$ then
        let $res = (pk_u, \mathsf{diff}(toll, cost))$ in $\mathsf{out}(ch, (res, \mathsf{sign}(sk_A, res)))$
      else if $\mathsf{diff}(toll, cost) <> \mathsf{diff}(toll, \mathsf{f}(rl))$ then
        let $res = (pk_u, \mathsf{diff}(\mathsf{diff}(toll, cost), \mathsf{diff}(toll, \mathsf{f}(rl))))$ in
        $\mathsf{out}(ch, (res, \mathsf{sign}(sk_A, res)))$.

Figure 5: The $commUA$ process.

5.1 ProVerif

ProVerif is a tool for automatic analysis of security protocols. It takes a protocol modelled in the applied pi calculus as input and then outputs whether this protocol satisfies a property which is described as a query. The output is sound but not complete. It is sound because if ProVerif claims that a property holds then the protocol does enforce the property. It is incomplete because it may neither be able to prove nor disprove a property for some protocols.

There are three types of queries supported by ProVerif which we will use to describe the security properties of GroupETP. *Reachability* is modelled as $query\ F$ where F is either $attacker(M)$ or $event(e(M_1, \ldots, M_k))$. The former means whether the adversary can construct M after the protocol is executed while the latter indicates that an event can happen. *Correspondence*, modelled as $query\ e \Rightarrow e'$, means "*if an event e has been executed, then the event e' has been previously executed*".

ProVerif uses a constructor $\mathsf{choice}[x, y]$ to define two processes P_1 and P_2 where x is evaluated in P_1 and y is evaluated in P_2. When choice is used, ProVerif will automatically verify the observational equivalence between the two processes without explicit property queries.

ProVerif has been used to verify a number of security protocols. With regard to location privacy, Dahl et al. manage to use ProVerif to analyse vehicular mix-zones [12] and another ETP system – VPriv [13]. ProVerif has also been applied on protocols in other application domains, such as certified email protocols [28], electronic cash protocols [29], online auction [30], electronic healthcare [31] and electronic voting [27, 32].

5.2 Correctness

Correctness means that the server can collect the right amount of tolls and all users exactly pay their tolls. Recall that there are two underlying assumptions according to practical toll pricing scenarios. One is that a user has no intention to pay more than his actual tolls while the other is that the server wants no loss of users' tolls. In GroupETP, this means that the server never throws away location records or decreases fees of any location records, and users never commit larger payments.

To prove the satisfaction of correctness, it is sufficient to prove that no users can pay less tolls. We can make use of a reachability assertion to verify whether the attacker can reach such an event after the protocol is finished. In the user process, the event happens after he receives res signed by the authority and when $\mathsf{add}(\mathsf{costU}(l), res) \neq \mathsf{f}(l)$ while function add models the addition operation on positive numbers. We use $event\ \mathsf{incorrect}(\mathsf{costU}(l), res)$ to denote this event. By inquiring

$event$ incorrect(costU(l), res), we can check the property of correctness. To implement this, we introduce a judge process (see Fig. 6 and add an extra operation before the payment process pay in the process $disputRes$. The user sends the resolution result res and his location l to the judge through a private channel pch, i.e., out(pch, (l, res, $cost$)). ProVerif gives a negative result, which means that the adversary cannot reach the incorrect event and GroupETP thus satisfies correctness.

$judge(rl, res, cost) \triangleq$in($pch$, ($rl$, res));
if add(res, $cost$)<>f(l) then
event incorrect($cost$, res).

Figure 6: The judge process.

5.3 Accountability

Accountability means that upon detection of a malicious behaviour, GroupETP can identify the originator of the behaviour. In GroupETP, the set of possible misbehaviours contains:

- e_1: users send smaller toll payment to the server;
- e_2: the server attaches wrong fees to location tuples;
- e_3: the server sends false location tuples to the authority;
- e_4: the server sends less toll payments to the authority.

Once any of the misbehaviours happens, the authority is always able to provide sufficient evidence to prove its identification of the originator.

Accountability is similar to the *non-repudiation* property which has been analysed in the literature [28] before. We can divide it into two sub-properties: (1) if a misbehaviour has been detected and a user has been found as the originator, then the user must have originated the malicious behaviour; (2) once a user has originated a misbehaviour, the system must be able to detect it and identify the user.

We use $event$ begin(e, pk(sk_u)) to denote that user u has generated a misbehaviour e and $event$ end(e, pk(sk_u)) to represent that the user u has been detected as the generator of the misbehaviour e. It has been addressed in [28] that the satisfaction of accountability also requires reliable channels between principles in the system. This means that a message sent on a channel should eventually reach its destination. To conclude, accountability is satisfied once the following two conditions hold for each misbehaviour e:

1. if end(e, pk(sk_u)) has been executed then begin(e, pk(sk_u)) has been previously executed;
2. if begin(e, pk(sk_u)) has been executed, then end(e, pk(sk_u)) will be executed in the future with reliable channels.

The first condition can be reduced to the correspondence assertion, end(e, pk(sk_u)) $\Rightarrow$ begin(e, pk(sk_u)) and we can have an automatic analysis of it using ProVerif. However, as ProVerif does not take reliable channels into account, we cannot verify the second condition by ProVerif. Hence, we do a manual proof for the second condition. Since the proof just follows the execution steps of the protocol, it is much simpler than the analysis of the first one, which should be go backwards through all execution histories leading to end(e, pk(sk_u)).

In GroupETP, misbehaviour may come from either users or the server. For the misbehaviour originated from the server, the first condition is trivial to verify. The behaviour of the server is unpredictable as it is part of the adversary. However, we can add the event begin(e', pk(sk_S)) after the authority receives the first two messages. This is because sending the two messages are the first possible behavior

once the server has misbehaved. In addition, the corresponding event $\mathsf{end}(e', \mathsf{pk}(sk_S))$ occurs at the later steps as the authority is responsible to catch the server's misbehaviour in GroupETP. Thus, the correspondence assertion $\mathsf{end}(e', \mathsf{pk}(sk_S)) \Rightarrow \mathsf{begin}(e', \mathsf{pk}(sk_S))$ is always true as they are executed sequentially in the same process.

In the following discussion, we take the most typical misbehaviour that a user cheats on his tolls as an example to show how an analysis for accountability looks like.

Automatic part of the proof. We first slightly modify the model of the user process by adding the event $\mathsf{begin}(e, \mathsf{pk}(sk_u))$. Recall that the value of $cheat$ is passed as parameters in the main process and if it is set to true then the user commits a smaller number of tolls. Hence, we add $event$ $\mathsf{begin}(e, \mathsf{pk}(sk_u))$ after $\mathsf{out}(ch, (\mathsf{pk}(sk_u), \mathsf{costU}(l), sig_u))$ in the user process.

In GroupETP, any misbehaviour is detected by the authority. If a user committed fewer tolls, the authority will determine the number of rest tolls in the sub-process communicating with the user $commUA$ by publishing the signed res on the public channel. Therefore, we add $event$ $\mathsf{end}(e, pk_u)$ after each of the two occurrences of $\mathsf{out}(ch, (res, \mathsf{sign}(sk_A, res)))$ in the process of $commUA$.

After giving the new model to ProVerif, we obtain that $\mathsf{end}(e, \mathsf{pk}(sk_u)) \Rightarrow \mathsf{begin}(e, \mathsf{pk}(sk_u))$ is true, which means that the correspondence assertion is satisfied.

Manual part of the proof. After the user u sends fewer tolls $\mathsf{costU}(l)$ to the server, the server will eventually receive the message $(\mathsf{pk}(sk_u), \mathsf{costU}(l), \mathsf{sign}(sk_u, \mathsf{costU}(l)))$ and initiates the dispute resolution protocol with the authority because it would lose money otherwise. Due to the same reason, the information including $\mathsf{h}(l)$, $\mathsf{gsign}(sk_u, \mathsf{h}(l))$ and $\mathsf{sign}(sk_u, \mathsf{costU}(l))$ will be included in the messages sent to the authority from the server. For the user, the amount of tolls the server asked for (i.e., $toll$) is also attached. The authority will receive this message and find the inequality between $toll$ and $\mathsf{costU}(l)$. In the sub-process $commUA$ with u, the authority will send the challenge message $(\mathsf{h}(l), toll, \mathsf{gsign}(sk_u, \mathsf{h}(l)))$ on the private channel which only u and the authority have access to. Then the user sends the real location l to the authority on the same channel, which will reach the authority. If the user does not send l, he will be considered having misbehaved and thus punished based on the punishment policy of the GroupETP system. The sub-process $commUA$ will eventually reach the event $\mathsf{end}(e, pk_u)$ as both of the conditions $toll = \mathsf{f}(rl)$ and $\mathsf{diff}(toll, cost) <> \mathsf{diff}(toll, \mathsf{f}(rl))$ are activated when $cost <> \mathsf{f}(rl)$, and at the moment $cost$ is $\mathsf{costU}(\mathsf{rl})$ which is obviously not equal to $\mathsf{f}(l)$.

5.4 Unlinkability

Unlinkability holds when, from the information learned from the execution of GroupETP, the attacker cannot decide whether a user has travelled on any location. In other words, the link between a user and his location records should be hidden from the adversary.

In GroupETP, the authority is assumed honest but curious, which means that the authority follows the protocols and does not collude with the adversary. *Conditional unlinkability* is satisfied with regards to the authority. If the server does not manipulate the fees in $\mathcal{S}$, the authority will learn nothing about the owner of any location but the number of each user's location tuples. This is due to the properties of the hash function and the randomness of location tuples (random numbers added). Whereas, if the server cheats on the fees of some tuples, then the affected users have to reveal those location tuples. This allows the authority to learn their owners as a result. As the choice of the revealed locations cannot be controlled by the authority, our system enforces conditional unlinkability.

Considering that the server is part of the adversary, we can make use of observational equivalence to define unlinkability. First we have to suppose there are at least two users. Otherwise, there is no privacy at all. Furthermore, these two users have to be honest, meaning that they always commit their

right amount of tolls. This is because GroupETP only protects honest users' privacy. Similar to [13], we suppose the locations of these two users have the same fees according to the charging policy. This is because the user process models a user with only one location record and in practice, a user's committed payment does not leak any information about his locations. The intuition behind unlinkability is that if any two users swap a pair of locations, the adversary cannot observe the difference. The main process P can then be described by Fig. 7. Let loc_1 and loc_2 be two location records. Then we say that GroupETP satisfies unlinkability if the following observational equivalence holds:

$$P[loc_1/l_1, loc_2/l_2] \approx P[loc_2/l_1, loc_1/l_2].$$

Our model can be automatically verified by ProVerif and the result is positive, which means GroupETP satisfies unlinkability.

$P \triangleq$ new sk_S; new sk_A; new sk_{u_1}; new sk_{u_2};
new l_1; ...; new l_2;
let $pk_A = \mathsf{pk}(sk_A)$ in
let $pk_{u_1} = \mathsf{pk}(sk_{u_1})$ in
let $pk_{u_2} = \mathsf{pk}(sk_{u_2})$ in
$\mathsf{out}(ch, sk_S) \mid \mathsf{out}(ch, pk_A) \mid \mathsf{out}(ch, pk_{u_1}) \mid \mathsf{out}(ch, pk_{u_2}) \mid$
let $l_1 = \mathsf{choice}[loc_1, loc_2]$ in
let $l_2 = \mathsf{choice}[loc_2, loc_1]$ in
$User(sk_{u_1}, pk_A, \mathsf{false}, l_1) \mid User(sk_{u_m}, pk_A, \mathsf{false}, l_2) \mid !Authority(pk_S, sk_A)$

Figure 7: The main process for unlinkability.

We now give the main theorem addressing that our ETP system satisfies the defined properties.

Theorem 1. *Our ETP system satisfies correctness, accountability and unlinkability.*

5.5 Secrecy & authentication

The protocols of each phase have to satisfy some basic properties such as secrecy and authentication. Such properties are required to be enforced under the assumption that all participants of the protocols are honest. Thus, we model the protocols of each phase individually and check whether they are secure in terms of secrecy and authentication. As the verification of authentication and secrecy is common in formal analysis and supported well by ProVerif, we do not give the models of the protocols and refer readers to [26] for the ProVerif codes. The results are given in Tab. 1.

Table 1: Verification of authentication and secrecy.

Protocols	Authentication		Secrecy
	injective	non-injective	
setupUS	–	u & S	pin
setupUA	A	u	sn
Toll Calculation	–	u & S	$cost_u$
Dispute resolving	S & A	–	$\mathcal{S}_u, \mathcal{R}'_u, r_u$

We use setupUS to denote the protocol between the server (S) and a user (u) in the setup phase and setupUA is between a user and the authority (A). We say a term is secret if the attacker cannot get

it by eavesdropping and sending messages, and performing computations [14]. For authentication, we consider two notions, namely, *agreement* and the slightly stronger notion *injective agreement* [33]. Agreement roughly guarantees to an agent A that his communication partner B has run the protocol as expected and that A and B agreed on all exchanged data values. Injective agreement further requires that each run of A corresponds to a unique run of B.

6 Conclusion

In this paper, we proposed a simple design of the GroupETP system, which preserves users' anonymity within groups [1]. Moreover, we resorted to formal methods to prove its security in terms of correctness, accountability and unlinkability. We modelled GroupETP using the applied pi calculus and, with the help of ProVerif, we successfully verified the system. From the analysis results, we confirmed that in addition to a balance between users' privacy and communication and computation overhead, the system also enforces the claimed security properties.

In future, we plan to develop a prototype of our system and conduct experiments to evaluate different group management policies. There are other ETP systems proposed recently which have some good properties. For instance, the ETP system Milo [11] provides techniques based on blind identity based encryption to strengthen spot checks in PrETP [9] in order to protect against large-scale driver collusion. It is interesting to see how to adopt these techniques into our system.

References

[1] X. Chen, G. Lenzini, S. Mauw, and J. Pang, "A group signature based electronic toll pricing system," in *Proc. ARES*. IEEE CS, 2012, pp. 85–93.

[2] J. Currie and R. Walker, "Traffic congestion and infant health: Evidence from E-ZPass," National Bureau of Economic Research, Working Paper 15413, 2009.

[3] C. Troncoso, G. Danezis, E. Kosta, and B. Preneel, "PriPAYD: Privacy friendly pay-as-you-drive insurance," in *Proc. WPES*. ACM Press, 2007, pp. 99–107.

[4] F. Giannotti, M. Nanni, D. Pedreschi, F. Pinelli, and M. Axiak, "Trajectory pattern mining," in *Proc. SIGKDD*. ACM Press, 2007, pp. 330–339.

[5] Y. Zheng, L. Zhang, X. Xie, and W.-Y. Ma, "Mining interesting locations and travel sequences from gps trajectories," in *Proc. WWW*. ACM Press, 2009.

[6] X. Chen, J. Pang, and R. Xue, "Constructing and comparing user mobility profiles for location-based services," in *Proc. SAC*. ACM Press, 2013, pp. 264–269.

[7] W. de Jonge and B. Jacobs, "Privacy-friendly electronic traffic pricing via commits," in *Proc. FAST*, ser. LNCS, vol. 5491. Springer, 2008, pp. 143–161.

[8] R. A. Popa, H. Balakrishnan, and A. J. Blumberg, "VPriv: Protecting privacy in location-based vehicular services," in *Proc. USENIX Security Symposium*. USENIX Association, 2009, pp. 335–350.

[9] J. Balasch, A. Rial, C. Troncoso, and C. Geuens, "PrETP: Privacy-preserving electronic toll pricing," in *Proc. USENIX Security Symposium*. USENIX Association, 2010, pp. 63–78.

[10] F. Garcia, E. Verheul, and B. Jacobs, "Cell-based roadpricing," in *Proc. EuroPKI*, ser. LNCS, vol. 7163. Springer, 2011, pp. 106–122.

[11] S. Meiklejohn, K. Mowery, S. Checkoway, and H. Shacham, "The phantom tollbooth: Privacy-preserving electronic toll collection in the presence of driver collusion," in *Proc. USENIX Security Symposium*. USENIX Association, 2011.

[12] M. Dahl, S. Delaune, and G. Steel, "Formal analysis of privacy for vehicular mix-zones," in *Proc. ESORICS*, ser. LNCS, vol. 6345. Springer, 2010, pp. 55–70.

[13] ——, "Formal analysis of privacy for anonymous location based services," in *Proc. TOSCA*, ser. LNCS, vol. 6993. Springer, 2011, pp. 98–112.

[14] B. Blanchet, "An efficient cryptographic protocol verifier based on prolog rules," in *Proc. CSFW*. IEEE CS, 2001, pp. 82–96.

[15] D. Dolev and A. C.-C. Yao, "On the security of public key protocols," *IEEE Transactions on Information Theory*, vol. 29, no. 2, pp. 198–207, 1983.

[16] B. Hoh, M. Gruteser, H. Xiong, and A. Alrabady, "Enhancing security and privacy in traffic-monitoring systems," *IEEE Pervasive Computing*, vol. 5, no. 4, pp. 38–46, 2006.

[17] ——, "Preserving privacy in GPS traces via uncertainty-aware path cloaking," in *Proc. CCS*. ACM Press, 2007, pp. 161–171.

[18] R. Shokri, G. Theodorakopoulos, J.-Y. L. Boudec, and J.-P. Hubaux, "Quantifying location privacy," in *Proc. S&P*. IEEE CS, 2011.

[19] J. Krumm, "Inference attacks on location tracks," in *Proc. Pervasive*, ser. LNCS, vol. 4480. Springer, 2007, pp. 127–143.

[20] D. Chaum and E. van Heyst, "Group signatures," in *Proc. EUROCRYPT*, ser. LNCS, vol. 547. Springer, 1991, pp. 257–265.

[21] M. Bellare, H. Shi, and C. Zhang, "Foundations of group signatures: The case of dynamic groups," in *Proc. CT-RSA*, ser. LNCS, vol. 3376. Springer, 2005, pp. 136–153.

[22] B. Przydatek and D. Wikström, "Group message authentication," in *Proc. SCN*, ser. LNCS, vol. 6280. Springer, 2010, pp. 399–417.

[23] M. Abe, S. S. M. Chow, K. Haralambiev, and M. Ohkubo, "Double-trapdoor anonymous tags for traceable signatures," in *Proc. ACNS*, ser. LNCS, vol. 6715. Springer, 2011, pp. 183–200.

[24] J. Guo, J. P. Baugh, and S. Wang, "A group signature based secure and privacy-preserving vehicular communication framework," in *Proc. INFOCOM Workshops*. IEEE CS, 2007, pp. 103–108.

[25] M. Abadi and C. Fournet, "Mobile values, new names, and secure communication," in *Proc. POPL*. ACM Press, 2001, pp. 104–115.

[26] X. Chen, G. Lenzini, S. Mauw, and J. Pang, "The ProVerif codes for the group signature based toll pricing system," available at http://satoss.uni.lu/xihui/publications.php/groupETP.zip.

[27] S. Kremer and M. D. Ryan, "Analysis of an electronic voting protocol in the applied pi calculus," in *Proc. ESOP*, ser. LNCS, vol. 3444. Springer, 2005, pp. 186–200.

[28] M. Abadi and B. Blanchet, "Computer-assisted verification of a protocol for certified email," *Science of Computer Programming*, vol. 58, no. 1-2, pp. 3–27, 2005.

[29] L. Luo, X. Cai, J. Pang, and Y. Deng, "Analyzing an electronic cash protocol using applied pi-calculus," in *Proc. ACNS*, ser. LNCS, vol. 4521. Springer, 2007, pp. 87–103.

[30] N. Dong, H. L. Jonker, and J. Pang, "Analysis of a receipt-free auction protocol in the applied pi calculus," in *Proc. FAST*, ser. LNCS, vol. 6561. Springer, 2011, pp. 223–238.

[31] N. Dong, H. Jonker, and J. Pang, "Formal analysis of privacy in an eHealth protocol," in *Proc. ESORICS*, ser. LNCS, vol. 7459. Springer, 2012, pp. 325–342.

[32] M. Backes, C. Hritcu, and M. Maffei, "Automated verification of remote electronic voting protocols in the applied pi-calculus," in *Proc. CSF*. IEEE CS, 2008, pp. 195–209.

[33] G. Lowe, "A hierarchy of authentication specification," in *Proc. CSFW*. IEEE CS, 1997, pp. 31–44.