A Group Signature Based Electronic Toll Pricing System

Xihui Chen*, Gabriele Lenzini*, Sjouke Mauw*†, Jun Pang†
*Interdisciplinary Centre for Security Reliability and Trust, University of Luxembourg, Luxembourg
†Faculty of Science, Technology and Communication, University of Luxembourg, Luxembourg

***Abstract*—With the prevalence of GNSS technologies, nowadays freely available for everyone, location-based vehicle services such as electronic tolling pricing systems and pay-as-you-drive services are rapidly growing. Because these systems collect and process travel records, if not carefully designed, they can threaten users' location privacy. Finding a secure and privacy-friendly solution is a challenge for system designers. Besides location privacy, communication and computation overhead should be taken into account as well in order to make such systems widely adopted in practice. In this paper, we propose a new electronic toll pricing system based on group signatures. Our system preserves anonymity of users within groups, in addition to correctness and accountability. It also achieves a balance between privacy and overhead imposed upon user devices.**

I. 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. Since Norway built the first working ETP system in 1986, ETP systems have been implemented worldwide. Nowadays, by exploiting the availability of free Global Navigation Satellite Systems (GNSS), traditional ETP systems are evolving into more sophisticated location-based vehicular services. They 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) [1]. Moreover, the collected traffic usage records can be used for public interest, such as planning roads' maintenance, or resolving legal disputes in case of accidents. As location is usually considered as a sensitive and private piece of information, ETP and PAYD systems raise obvious privacy concerns. In addition, by processing locations and travel records, they can learn and reveal users' sensitive information such as home addresses and medical information [2], which consequently can lead to material loss or even bodily harm. Building secure ETP and PAYD systems that guarantee *location privacy* and high quality of service is actually a scientific challenge.

In the last few years, secure ETP and PAYD systems have been widely studied [1], [3], [4], [5], [6], [7]. They can roughly be divided into two categories based on whether locations are stored in user devices or collected by central toll servers. PriPAYD [1], PrETP [5], the cell-based solution described in [6], and Milo [7] 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 [4], where the server stores a database of users' travel history, and the ETP system described in [3], where the server collects the hash values of trip records.

Both categories have advantages and disadvantages. Hiding locations from servers drastically reduces the concerns about location privacy. However, the load for user devices is considerable. Typically, devices have to manage the storage of locations and proofs which convince servers that they have not cheated, e.g., making use of zero-knowledge proofs. On the other side, the availability of location databases collected by servers, e.g., in VPriv, can help improve applications such as traffic monitoring and control although the integration of multiple systems should be carried out carefully. Whereas, offering preservation of location privacy become a mandatory requirement. In this paper, we follow the design principles of VPriv [4].

In VPriv, users select a set of random tags beforehand and send their locations attached with these tags to the toll server. The server then computes and returns *all* location fees. Each user adds up his location fees according to his tags and proves the summation's correctness to the server by using zero-knowledge proof, without revealing the ownership of the tags. This process needs to run several rounds to avoid user behaviours deviating from the system. Thus the main disadvantage with VPriv is that the computation and communication overhead increases linearly with the number of *rounds* executed and with the number of *users*.

Our contributions. We propose a novel but simple ETP system which achieves a balance between privacy and overhead for users. By dividing users into groups and calculating tolls in one round, we reduce the amount of exchanged information as well as the computation overhead due to the smaller number of locations of a group. We use group signature schemes to guarantee anonymity within a group, with an authority being the group manager. Note that the concept of groups, however, requires us to design an effective group division policy to optimally preserve users' location privacy (discussed in Sect. VI). We have proved that our system is correct, which guarantees that users always pay their usage to the server, and assures accountability, which guarantees originators of misbehaviours can always be found. Moreover, our system is also proved to be able to enforce conditional unlinkability between users and their locations.

Structure of the paper. Sect. II describes the participants of our system, the threat model and the assumptions, and also states the security goals of our design. Sect. III recalls group signature schemes and other cryptographic primitives we adopt. Our ETP system is fully described in Sect. IV. Sect. V defines the security properties and shows their enforcement by our system. We conclude our paper in Sect. VI with some discussions and ideas for future work.

II. System Model

A. Principals

The system consists of four principals: *users*, *their cars* with on board units (*OBUs*), *the authority*, and *the toll server* (see Fig. 1).

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, 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.

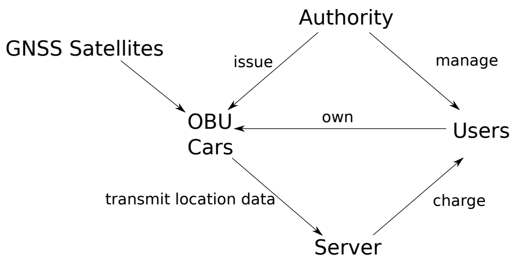

Fig. 1. The relationship among the principals.

B. 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.

C. Assumptions

Considering the practice of ETP systems, we make a general assumption that users of such systems tend to pay less tolls and the server wants no economic loss (*Assumption 1*). For instance, in our system 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 [8] (*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 [9], 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 [10] or taking into account users' mobility profiles [11], and users' private information can thus be inferred [12]. However, we observed from 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 traveling users is not small. Therefore, similar to VPriv, in this paper, we focus on privacy leakage from ETP systems without considering the above mentioned techniques and attacks (*Assumption 5*).

D. Security Properties

In addition to reducing communication and computation overheads, our system should employ proper measures to protect honest users and servers. Referring to what other systems achieve (e.g., [1], [4], [5]), we address the following security properties, their formal definitions are given in Sect. V. :

Correctness. Clients pay their road usage and the server collects 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. An intruder cannot link a given location record to its generator.

III. Cryptographic Primitives

Group Signature Schemes. Group signatures [13] 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

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

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.

- CORRECTNESS Signatures produced by a group 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 [14]).

In the last decade, efficient group signature schemes with new fancy features have been developed, e.g., group message authentication [15] and group signcryption [16]. 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.

IV. THE ETP SYSTEM

A. 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. VI). 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.

B. Notations

Tab. I summarises the important notations. With c, S, and A we indicate a user, the server, and the authority, respectively. With $f(\ell, t)$ we indicate the fee to be paid when passing location ℓ at time t, while $cost_c$ is the amount of fees that c committed to pay after the toll session sid. We use $Sig_c(m)$ to denote the signature on message m signed by c, and $Gs_c(m)$ denotes the group signature of c on message m. For other cryptographic primitives, we use standard notations.

TABLE I
NOTATIONS.

$f(\ell, t)$	The fee of passing position ℓ at time t
$cost_c$	The committed toll payment of user c
$toll_c$	The amount of tolls of user c computed based on the fees the server calculates
$\mathcal{R}_c$	The set of location tuples of user c
$\mathcal{R}$	The set of dispute solutions
$\mathcal{L}$	The set of location tuples that the server has collected
sid	The identifier of the toll session
$Sig_X(m)$	Signature of message m generated by a principal X
$Gs_c(m)$	Group signature of message m generated by a group member c
$gpk(\mathcal{G})$	The group public key of group $\mathcal{G}$
$pk(X)$	The public key of a principal X
$sk(X)$	The private key of a principal X
$h(m)$	The *hash* value of message m
$Enc_{pk(X)}(m)$	The message m encrypted with X's public key $pk(X)$

C. 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.

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 c as an example. Let pin be his pin code and sn the serial number. The Set-up protocol is depicted in Fig. 2. Upon receiving the user's public key, the server checks c's signature on pin. If valid, the server replies with its signature on the key, which the user sends to the authority subsequently when joining a group. A replay attack on c's message to the authority is not feasible, as the same group would be returned if the same request message arrives again. Fig. 2 does not include 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.

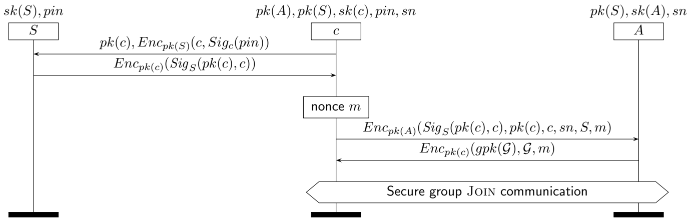
$sk(S), pin$
$pk(A), pk(S), sk(c), pin, sn$
$pk(S), sk(A), sn$
S
c
A
$pk(c), Enc_{pk(S)}(c, Sig_c(pin))$
$Enc_{pk(c)}(Sig_S(pk(c), c))$
nonce m
$Enc_{pk(A)}(Sig_S(pk(c), c), pk(c), c, sn, S, m)$
$Enc_{pk(c)}(gpk(\mathcal{G}), \mathcal{G}, m)$
Secure group JOIN communication

Fig. 2. The *Set-up* protocol.

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. A message from user c, a member of group $\mathcal{G}$, is denoted by $(\langle \ell, t, r, \mathcal{G} \rangle, Gs_c(h(\ell, t, r))$. After receiving this message, the server verifies $Gs_c(h(\ell, t, r))$ 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. With $\mathcal{R}_c$ denoting the locations which c has travelled and are stored on the USB stick, we depict the protocol in Fig. 3.

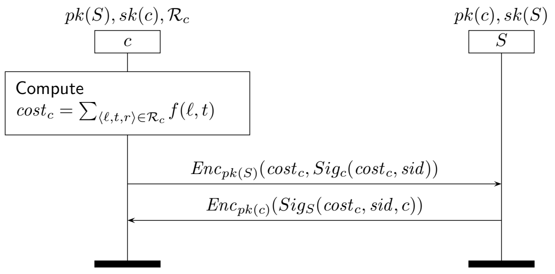
$pk(S), sk(c), \mathcal{R}_c$
$pk(c), sk(S)$
c
S
Compute
$cost_c = \sum_{\langle \ell, t, r \rangle \in \mathcal{R}_c} f(\ell, t)$
$Enc_{pk(S)}(cost_c, Sig_c(cost_c, sid))$
$Enc_{pk(c)}(Sig_S(cost_c, sid, c))$

Fig. 3. The *Toll Calculation* protocol.

The user starts with calculating his toll payment $cost_c$ in toll session sid. For each $\langle \ell, t, r \rangle$ in $\mathcal{R}_c$, the user computes its fee $f(\ell, t)$ according to the server's public charging policy and obtains $cost_c$ by adding all tuple fees up. Then the user sends to the server his signature on $cost_c$ and session identifier sid, which indicates that user c's toll payment in toll session sid is $cost_c$. After receiving c's message, the server verifies the signature before sending back its signature on c'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.

Phase 4: Dispute Resolving. In this protocol, with the help of the authority, the server finds cheating users and the amount of tolls unpaid. 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

$$\sum_{c \in \mathcal{G}} cost_c < \sum_{\langle \ell, t, r \rangle \in \mathcal{L}} f(\ell, t)$$

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:

$$\mathcal{S} = \{\langle h(\ell, t, r), f(\ell, t), G_{\mathcal{S}_c}(h(\ell, t, r)) \rangle \mid \forall (\ell, t, r) \in \mathcal{L}, \forall c \in \mathcal{G}\}$$

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

$$\mathcal{T} = \{\langle c, cost_c, Sig_c(cost_c, sid) \rangle \mid \forall c \in \mathcal{G}\}$$

Subsequently, the server constructs a message consisting of $\mathcal{S}$,$\mathcal{T}$ and $Sig_S(\mathcal{S}, sid)$, and sends it to the authority. 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. III. 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., $toll_c$) is computed in lines 16-17. When it is not larger than the user's committed one (i.e., $cost_c$), 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 users are also possible to have committed smaller payments. For any user c with $toll_c > cost_c$, there is a pair $(c, toll_c - cost_c)$ in the result res. Moreover, all corresponding tuples of user c in $\mathcal{S}$ are stored in set $\mathcal{S}_c$.

Algorithm 1 Function $SvrDisRes$

```
 1: Input: $\mathcal{S}$, $\mathcal{T}$, $signS$
 2: Output: $res$
 3: $res := \emptyset$;    $\mathcal{S}_c = \emptyset$;    $toll_c := 0$;
 4: for all $(c, cost, sign) \in \mathcal{T}$ do
 5:    if $checksign(sign, (cost, sid), pk(c)) = false$ 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:          $c :=$OPEN$(gsign)$ ;
17:          $toll_c := toll_c + feeLoc$;
18:          $\mathcal{S}_c := \mathcal{S}_c \cup \{\langle hashLoc, feeLoc, gsign\rangle\}$;
19:    end if
20: end for
21: for all $toll_c > cost_c$ do
22:    $res := res \cup \{(c, toll_c - cost_c)\}$;
23: end for
24: return $res$
```

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 (in Fig. 4) with it. 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 Fig. 4.

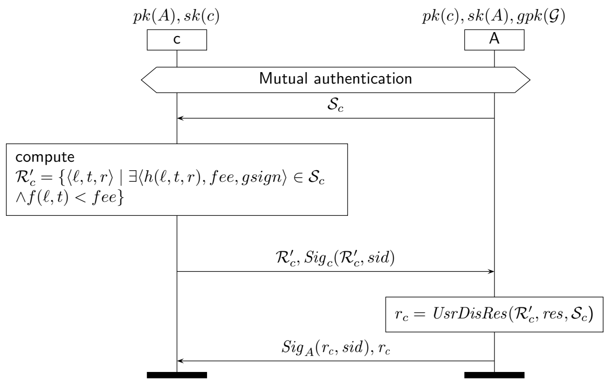
$pk(A), sk(c)$
$pk(c), sk(A), gpk(\mathcal{G})$
c
A
Mutual authentication
$\mathcal{S}_c$
compute
$\mathcal{R}'_c = \{\langle \ell, t, r\rangle \mid \exists\langle h(\ell, t, r), fee, gsign\rangle \in \mathcal{S}_c \wedge f(\ell, t) < fee\}$
$\mathcal{R}'_c, Sig_c(\mathcal{R}'_c, sid)$
$r_c = UsrDisRes(\mathcal{R}'_c, res, \mathcal{S}_c)$
$Sig_A(r_c, sid), r_c$

Fig. 4. The *User Dispute Resolving* protocol.

The user finds the set of location tuples $\mathcal{R}'_c \in \mathcal{R}_c$ with larger fees in $\mathcal{S}_c$, 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 $\mathcal{R}'_c$ by verifying the user's signature (line 4-5). Then for each location tuple $\langle \ell, r, t\rangle \in \mathcal{R}'_c$, the authority finds the tuple $\langle hashLoc, fee, gsign\rangle \in \mathcal{S}_c$ 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 c's real tolls. By subtracting the user's committed payment $cost_c$, we obtain the rest of tolls the user still needs to pay, i.e., r_c (line 10) which is called *dispute resolution* in the following discussion.

Algorithm 2 Function $UsrDisRes$

```
 1: Input: $\mathcal{R}'_c$, $signRc$, $\mathcal{S}_c$
 2: Output: $r_c$
 3: $\delta := 0$;
 4: if $checksign(signRc, (\mathcal{R}'_c, sid), pk(c)) = false$ then
 5:       return ’check of $\mathcal{R}'_c$ failed’;
 6: end if
 7: for all $\langle \ell, t, r\rangle \in \mathcal{R}'_c$ do
 8:    if $\exists\langle h(\ell, r, t), fee, gsign\rangle \in \mathcal{S}_c$ then
 9:          $\delta_c := \delta_c + (fee - f(\ell, t))$;
10:    end if
11: end for
12: $r_c := (c, toll_c - cost_c - \delta_c)$;
13: return $r_c$
```

For each user c, $r_c = (c, 0)$ indicates that the server has misbehaved and the user is innocent while $r_c = (c, toll_c - cost_c)$ means the server is honest and the user paid less. If $toll_c - cost_c - \delta_c < 0$, 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 $\mathcal{R} = \{(c, v) \mid v = toll_c - cost_c - \delta_c \wedge v > 0\}$, 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.

Theorem 1. *Let $\mathcal{S}'$ be the set containing location tuples sent by the server to the authority during dispute resolution. If the server wants no loss of users' tolls, then for all $(\ell, t, r) \in \mathcal{L}$ there exists $\langle h(\ell, t, r), fee, Gs_c(h(\ell, t, r))\rangle \in \mathcal{S}'$.*

V. Security Properties & Analysis

In this section we define precisely what we mean by correctness, accountability and unlinkability, and briefly discuss how our system satisfies each of them. The full proof of our main theorem is given in Appendix A.

A. Correctness

Correctness means that the server can collect the right amount of tolls and all users pay their tolls exactly. Recall that there are two underlying assumptions according to practical toll 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. For our system, this means that the server never throws away location tuples and decreases fees, and users never commit larger payments.

Let $\overline{cost}_c$ be the real amount of tolls that user c should pay and pay_c be the amount of tolls that user c actually pays to the server after Phase 4 of our system. Let $\overline{cost}_\mathcal{G} = \sum_{c\in\mathcal{G}} \overline{cost}_c$ be the real amount of tolls from all users and $pay_\mathcal{G} = \sum_{c\in\mathcal{G}} pay_c$ the amount of tolls that the server actually collects from the group $\mathcal{G}$ after Phase 4 of our system. With Theorem 1, the property of correctness can be defined as follows:

Definition 1 (Correctness)**.** *Suppose the server wants no loss of users' tolls and users have no intention to pay more than their tolls, then for any $c \in \mathcal{G}$ it holds that ($pay_c = \overline{cost}_c$), and for the server it holds that ($pay_\mathcal{G} = \overline{cost}_\mathcal{G}$).*

In our system, whenever a user has paid less, the server initiates the dispute resolving protocol with the authority who would give the correct toll of that user. Meanwhile, because of the properties of group signatures, e.g., Unforgeability and Exculpability, the server is unable to charge more locations than the ones users submitted.

B. Accountability

Accountability means that upon detection of malicious behaviour, our system can identify which principal has misbehaved.

Let $\mathcal{B}$ be the set of all potential misbehaviours from the attackers in our system. So relation $\mathcal{A} = \mathcal{B} \times \mathcal{U}$ represents all possible attacks and the corresponding attackers. In our system, $\mathcal{U} = \mathcal{C} \cup \{S, A\}$. Let $attacker : \mathcal{A} \to \mathcal{U}$ be the function mapping an attack to the attacker, e.g., $attacker((\beta, c)) = c$. Let E be the set of evidences during the run of our system and $\mathcal{P}(E)$ the power set of E. Thus, *accountability* is defined as follows:

Definition 2 (Accountability)**.** *Let $\mathcal{A}' \subseteq \mathcal{A}$ be the attacks that actually happen during the execution of our system in a toll session. For any $\alpha \in \mathcal{A}'$, our system is able to provide a set of evidences $E' \in \mathcal{P}(E)$ and there exists a function $\mathit{find} : \mathcal{P}(E) \times \mathcal{A} \to \mathcal{U}$ such that $\mathit{find}(E', \alpha) = attacker(\alpha)$.*

At steps where attackers may misbehave, our system provides sufficient evidence to find the originators. For instance, when the server did not send a user's toll payment to the authority on purpose, the server's signature on the user's payment could be taken as the evidence to prove the server's misbehaviour.

Despite the fact that our system assures accountability, resolving disputes is still a costly step. We can establish a proper punishment policy to discourage misbehaviours. This in turn also improves the efficiency and performance of our system. For instance, by punishing cheating users, the frequency of dispute resolving can actually be ensured to be very small in practice.

C. Unlinkability

Unlinkability holds when, from the information learned from the execution of our system, the attacker cannot decide whether a user has travelled on any location. In order to enforce this property, we should consider the following two aspects.

First, from all messages learned after the execution of our system, the attacker cannot link any location to its originator. For users, the properties of group signature schemes, i.e., Anonymity and Unlinkability guarantee this property with regards to the malicious server. With regards to the honest but curious authority, who does not collude with other attackers, *conditional unlinkability* is satisfied. If the server does not manipulate the fees in $\mathcal{S}$, the authority learns 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.

Second, the communication process of the system does not leak any information about linkability, meaning that the attacker cannot break unlinkability by analysing differences between executions of the system. To define unlinkability and check its satisfaction w.r.t. this situation, we make use of formal verification (see Appendix A).

We now give the main theorem addressing that our ETP system satisfies the defined properties. The full proof of the theorem is given in Appendix A.

Theorem 2. *Our ETP system satisfies correctness, accountability and unlinkability.*

We also verified secrecy and authentication, whose results are given in Appendix B.

VI. Discussion & Conclusion

In this paper, we have proposed a simple design for ETP systems which preserves users' anonymity within groups. The main goal of our system is to balance users' privacy with communication and computation overhead: a large group means better privacy for users, while this gives rise to more overhead when running the system. With the help of group signature schemes, our system is proved to guarantee correctness, accountability and unlinkability. To be complete, we still have the following issues to address.

Comparison with VPriv. As mentioned in Sect. I, our system resembles VPriv [4] that the server collects locations. However, VPriv imposes a relatively high overhead to users and the server. Compared with VPriv, in our system, the communication overhead between users and the server is reduced. Our system 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 in our system, namely the authority takes the responsibility to find misbehaved users or server. Hence, the server and users are released from a heavy computation 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' whereabout.

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 [17]. 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 enable users to change their group memberships, is also necessary. For instance, users move to another city or they are not satisfied with their current group. To find the optimal group size which can protect users' anonymity is part of our future work. Note that if a user has joined in multiple groups, the similarity between his travel records of these groups would decrease his anonymity.

Tamper resistant devices vs. spot checks. In order to ensure OBUs are not manipulated by users, e.g., to transmit false locations, we have to consider possible solutions. 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 there is one belonging to c. According to [4], a small number of spot checks with a high penalty would suffice.

Future work. In future, we plan to develop a prototype of our system and conduct experiments to evaluate different group management policies and to compare the efficiency of our system with PrETP and VPriv. A recently proposed ETP system Milo [7] provides techniques based on blind identity based encryption to strengthen spot checks in PrETP [5] to protect against large-scale driver collusion. It is interesting to see how to adopt these techniques into our system. Nowadays, automatic fare collection systems are increasingly used in public transport, which give user similar concerns on their privacy. It is also interesting to see whether we can extend the ideas in this paper to develop a new fare collection system.

Acknowledgement

We thank Flavio Garcia for his comments that helped to improve the paper.

References

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

[2] Z. Ma, F. Kargl, and M. Weber, "Measuring long-term location privacy in vehicular communication systems," *Computer Communications*, vol. 33, no. 12, pp. 1414–1427, 2010.

[3] 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.

[4] 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.

[5] 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.

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

[7] 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.

[8] 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.

[9] 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.

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

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

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

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

[14] 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.

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

[16] 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.

[17] 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.

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

[19] S. Delaune, S. Kremer, and M. D. Ryan, “Verifying privacy-type properties of electronic voting protocols,” *Journal of Computer Security*, vol. 17, no. 4, pp. 435–487, 2009.

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

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

Appendix

Appendix A: Proof of Theorem 2

A. Correctness

We prove correctness from the following two perspectives of users and the server.

We start proving that our system enforces that each user pays his real tolls, which is $\forall_{c\in\mathcal{G}}\ pay_c = \overline{cost_c}$. Assume that a user c has committed a smaller payment in Phase 3, i.e., $cost_c < \overline{cost_c}$. The server would find that $\sum_{c'\in\mathcal{G}} cost_{c'} < \sum_{\langle \ell,t,r\rangle\in\mathcal{L}} f(\ell,t)$ and thus construct and send $\mathcal{S}$ and $\mathcal{T}$ to the authority for solving the dispute. Then at least one of the following situation occurs depending on whether the server misbehaves.

Case 1. The server is honest and data $\mathcal{S}$, $\mathcal{T}$ are correct. Then the authority computes the set res by function $SvrDisRes$ which contains an item $(c, toll_c - cost_c)$ with $toll_c = \overline{cost_c}$. During the dispute resolving protocol with the authority, the user cannot provide any correct location tuple $\langle \ell, t, r\rangle$ that has a corresponding tuple in $\mathcal{S}_c$ with a larger fee. This is because all tuples in $\mathcal{S}_c$ are correct. Thus, the user's dispute resolution $r_c = (c, \overline{cost_c} - cost_c)$ is returned to the server, and the user has to pay the amount of tolls $pay_c = \overline{cost_c} - cost_c + cost_c$, equivalent to $\overline{cost_c}$.

Case 2. The server is dishonest. With the assumption that the server wants no loss of tolls, the set $\mathcal{S}_c$ has the same size with $\mathcal{R}_c$, and the set $\mathcal{R}'_c \in \mathcal{R}_c$ have a corresponding set of tuples $\mathcal{S}'_c \in \mathcal{S}_c$ with larger fees. Furthermore, we have $\overline{cost_c} = toll_c - \delta$ where $\delta = \sum_{\langle h(\ell,t,r), fee, gsign\rangle\in\mathcal{S}'_c}(fee - f(\ell,t))$. During the dispute resolution with the authority, user c identifies $\mathcal{S}'_c$ from $\mathcal{S}_c$ with the charging policy. As no users want to pay more than they should, a rational user sends $\mathcal{S}'_c$ back to the authority. Suppose the user's dispute resolution $r_c = (c, v)$, then $v = toll_c - cost_c - \delta$. Therefore, $pay_c = v + cost_c = \overline{cost_c}$.

We prove that $pay_{\mathcal{G}} = \sum_{c\in\mathcal{G}} \overline{cost_c}$ straightly from the following two facts: 1) Our above proof has shown that for each user $c \in \mathcal{G}$, $pay_c = \overline{cost_c}$; 2) $pay_{\mathcal{G}} = \sum_{c\in\mathcal{G}} pay_c$.

B. Accountability

In our system, the set $\mathcal{B}$ consists of the following misbehaviours:

- β_1: dishonest users send smaller toll payment to the server;
- β_2: malicious users refuse to pay tolls;
- β_3: the server attaches wrong fees to location tuples;
- β_4: the server sends false location tuples to the authority;
- β_5: the server sends less toll payments to the authority;

We prove accountability is secured against misbehaviours as described in the set $\mathcal{B}$.

Case 1. Assume $\alpha = (\beta_1, c)$ happens, i.e., $cost_c < \overline{cost_c}$. From the proof of correctness, we know the authority would have a dispute resolution $(c, \overline{cost_c} - cost_c)$ in $\mathcal{R}$. From the authority's signature on $\mathcal{R}$, the server learns that user c has given a smaller payment. Together with user c's signature on $cost_c$ and $\mathcal{R}'_c$, and $Sig_S(\mathcal{S}, sid)$, c cannot deny his misbehaviour.

Case 2. Assume (β_2, c) happens. We have the following two evidences – (i) the server does not have user c's signature on $cost_c$; (ii) user c cannot provide the server's signature on $cost$, i.e., $Sig_S(cost_c, sid, c)$.

Case 3. Assume (β_3, S) happens. In this attack, we have at least one tuple in $\mathcal{S}$ with a higher fee. Let it be $\langle h(\ell,t,r), fee, gsign\rangle$ where $fee > f(\ell,t)$, and the location $\langle \ell, t\rangle$ belongs to user c. When c receives $\mathcal{S}_c$, he identifies the tuple by comparing $h(\ell,t,r)$ and adds $\langle \ell,t,r\rangle$ to $\mathcal{R}'_c$. With the server's signature on $\mathcal{S}$ and the unforgeability of $gsign$, the user can prove to the authority that the server originates the attack.

Case 4. Assume (β_4, S) happens. With the properties Unforgeability, Exculpability and Coalition-resistance of group signature schemes, the server cannot forge any location with a correct group signature from any honest user. Therefore, a failure of function Verify (line 11 in Alg. 1) will indicate the server's misbehaviour.

CASE 5. Assume (β_5, S) happens. Suppose the server omits user c's payment, i.e., $cost_c$, which has been committed to the server during toll calculation. In this case, the authority would execute the user dispute solving protocol with the user. As c has the server's signature on $cost_c$ (i.e., $Sig_S(cost_c, sid, c)$), he can prove to the authority the server's omission of his payment.

C. Unlinkability

For attacks on analysis of messages used during the execution of the system, we prove that no significant information about linkability is learnt by a malicious server and a curious authority. First, we examine the information that the server can obtain from the system.

CASE 1. The set of location tuples summarised in $\mathcal{S}$ during the driving phase. Because of the anonymous channel and the properties of ANONYMITY and UNFORGEABILITY of group signatures, the server cannot link any location to its originator.
CASE 2. The set of users' committed payments summarised in $\mathcal{T}$. As the attack based on partitioning is not feasible, users' payments do not reveal any relationship between users and their locations.
CASE 3. The set of dispute resolution summarised in $\mathcal{R}$. The server only learns the tolls that misbehaved users have not paid. Nothing about specific locations is revealed.

Second, we examine the information the authority can obtain.

CASE 1. The set $\mathcal{S}$ and $\mathcal{T}$. Due to the preimage resistance of hash primitives, the authority cannot learn users' locations through opening location signatures.
CASE 2. The set $\mathcal{R}'_c$ of user c if the server increases the fee of any location tuple of user c. The authority learns the user has travelled the locations in $\mathcal{R}'_c$ but the authority cannot choose the locations, which instead are determined by the server.
With the assumption of no collusion between the server and the authority, *conditional unlinkability* is satisfied.

The second type of attacks on unlinkability is through observing the difference between system executions where users' locations are varied. We start from defining unlinkability w.r.t. this type of attacks and proceed proving our system's security using automatic formal verification. We use processes to denote participants' behaviours in protocols. Let $C\langle\varphi\rangle$ be the process representing user c originating location record φ and let A be the process of the authority. We use $C\langle\varphi\rangle \mid A$ to represent the parallel composition of these two processes, which admits all possible communications and interleavings. The intuition behind unlinkability is that if any two users swap a pair of locations, the adversary cannot observe the difference. Recall that the attacks on unlinkability exploring fees and users' payments are not feasible in practice. The 'difference' does not include the changes of users' payments after swapping locations. *Observational equivalence*, which defines indistinguishability between two processes [18], gives us an effective way to formalise unlinkability in our system. Similar to [19], we need at least two traveling users. Otherwise, an intruder can easily link all location tuples to one user.

Definition 3 (Unlinkability)**.** *Assume that $\mathcal{G}$ is a group with at least two users c and c' and assume any two location records φ and φ'. Unlinkability between a location tuple and its generator holds if*

$$A \mid C\langle\varphi\rangle \mid C'\langle\varphi'\rangle \approx A \mid C\langle\varphi'\rangle \mid C'\langle\varphi\rangle$$

Def. 3 is formulated against the malicious server. Similarly, conditional unlinkability against the authority can be defined by replacing the property with $C\langle\varphi\rangle \mid C'\langle\varphi'\rangle \approx C\langle\varphi'\rangle \mid C'\langle\varphi\rangle$, where φ and φ' are not in $\mathcal{R}'_c$. ProVerif [20] is an efficient tool for verifying security properties. It takes a protocol modelled as a process in the applied π calculus [18] as input and checks if the protocol satisfies a given property. As observational equivalence can be modelled and verified by ProVerif, unlinkability is able to be automatically verified. We have modelled our system and the unlinkability property, and got positive results from ProVerif. (ProVerif codes are available on request.)

APPENDIX B: VERIFICATION OF SECRECY AND AUTHENTICATION

We use ProVerif [20] to formally prove that our system as a whole does not suffer from attacks on security and authentication. The results are listed in Tab. II.

TABLE II
VERIFICATION OF AUTHENTICATION AND SECRECY.

Protocols	Authentication		Secrecy
	injective	non-injective	
setupCS	–	c & S	pin
setupCA	A	c	sn
Toll Calculation	–	c & S	$cost_c$
Dispute resolving	S & A	–	$\mathcal{S}_c, \mathcal{R}'_c, r_c$

We use setupCS to denote the protocol between the server (S) and a user (c) in the setup phase and setupCA 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 [20]. For authentication, we consider two notions, namely, *agreement* and the slightly stronger notion *injective agreement* [21]. 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. (ProVerif codes are available on request).