IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received April 22, 2020, accepted May 21, 2020, date of publication May 29, 2020, date of current version June 10, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2998504

A Construction for Providing Reusability to Mobile Phone-Based e-Tickets

RICARD BORGES AND FRANCESC SEBÉ
Department of Mathematics, Universitat de Lleida, 25001 Lleida, Spain
Corresponding author: Francesc Sebé (francesc.sebe@udl.cat)

This work was supported in part by the Spanish Ministry of Science, Innovation, and Universities, under Project MTM2017-83271-R.

ABSTRACT Nowadays, the use of electronic tickets in public transport is a reality. Several mobile phone-based e-ticket systems have been proposed so far. Even so, very few of them include reusability in the sense that a single e-ticket allows to make several journeys. Although the identity of users is usually hidden behind a pseudonym, the existing proposals providing reusability allow the system to link all the journeys made with a given e-ticket. In this paper we present a privacy-preserving construction allowing to endow a mobile phone-based e-ticket system with reusability. The privacy of users is proven to be preserved even assuming an internal attacker with full access to all the information managed by the system servers. All the sensitive interactions of a user with the system keep anonymous and unlinkable. Further, as a result of an inspection, the system is only able to determine whether the inspected user is allowed to make the current journey.

INDEX TERMS Cryptographic protocols, data privacy, information security, intelligent transportation systems, mobile communication.

I. INTRODUCTION

In 1662, the first bus service was deployed in Paris with seven horse-drawn vehicles running along regular routes [1]. Since then, public transport systems have evolved continuously. In the digital era, the world-wide deployment of Internet together with the popularization of mobile phones have set the basis for the development of advanced technology for the management of public transport. As a very relevant example, we mention the replacement of paper tickets with electronic ones (*e-tickets*).

Big data is a branch of computer science focused on the collection, storage, processing, and analysis of large amounts of data that frequently originate from disparate sources [2]. The collection and analysis of data about public transport allows to optimize the number of vehicles available in each time frame, anticipate an excess or lack of service due to exceptional events, or modify the routes adapting them to the dynamics of the city.

From the data privacy point of view, an accurate analysis of data about the particular way in which a person uses public transport allows to infer personal information such as her work schedule, hobbies, or even health problems. Legislation, like the EU General Data Protection Regulation [3], protects citizens from the non-authorized processing of personal data collected by any organization. Nevertheless, legislation can not protect people against processing of data with unlawful purposes in situations in which data has been stolen as a result of a cyberattack or after being leaked by an internal dishonest employee. So as to reduce such risks, companies should guarantee that the information they collect is restricted to that strictly necessary to provide the service requested by their customers. In the particular case of public transport e-tickets, the information managed by a ticket issuer must not allow to trace the way a specific citizen uses public transport.

A. E-TICKET SYSTEMS

A *ticket* is defined as *"a small piece of paper or card given to someone, usually to show that they have paid for an event, journey, or activity"* [4]. The definition for *e-ticket* is *"a ticket, usually for someone to travel on an aircraft, that is held on a computer and is not printed on paper"* [4].

Many e-ticket systems [5]–[9] consider a system model composed of three actors in which *users* make use of a service offered by a *service provider* through an e-ticket generated and managed by the *ticket issuer*. The service provider and ticket issuer roles may be taken by the same entity.

The associate editor coordinating the review of this manuscript and approving it for publication was Yassine Maleh.

There exist proposals in which e-tickets can be used anonymously [6]–[8], [10]–[15]. Some of them [6], [8], [11], [13], [15] include a *trusted third party* responsible for managing anonymity, whereas in other proposals [7], [10], [12], [14] anonymity is managed by the ticket issuer itself. Additionally, there may exist the so-called *inspection authorities* responsible for face-to-face checks [16].

Most proposals agree upon the existence of the following three main phases along the lifetime of an e-ticket [17]:

1) *Payment*: the user pays for the required service;
2) *Issuance:* the ticket issuer, after receiving an appropriate payment, generates an e-ticket that is provided to the user;
3) *Validation:* before granting the user with access to the service, the service provider verifies the validity of the presented e-ticket.

Some proposals consider *payment* and *issuance* as being part of the same phase [10], [12], [18], [19], whereas some others include an *anonymity revocation* procedure which allows to determine the identity of users who have acted fraudulently [6], [7], [10]–[14]. An additional *random inspection* process performed by inspection authorities is also considered in some proposal [16].

An e-ticket system must guarantee, or at least consider, the security and privacy requirements enumerated in Table 1 [17]. *Reusability* allows an e-ticket to be used for several journeys during a period of time.

TABLE 1. Security and privacy requirements for e-ticket systems.

Requirement	Description
Guarantee	
Integrity	An e-ticket can not be modified.
Authenticity	A user must be able to validate that an e-ticket has been issued through an authorized provider.
Non repudiation	The service provider can not deny having issued an e-ticket.
Unforgeability	Only authorized entities can issue a valid e-ticket.
Non overspending	An e-ticket can only be used the number of times stipulated at the time of its issuance.
Exculpability	The service provider can not falsely accuse an honest user.
Consider	
Identifiable e-ticket	The identity of the e-ticket owner can be verified.
Anonymity	
Fully-Revocable	Anonymity of the ticket owner can be lifted.
Selective-Revocable	The identity of the ticket owner can be revealed in case of a malicious act.
Anonymous	The identity of ticket owners can not be determined in any way.
Transferability	An e-ticket can be transferred to another user.
Reusability	An e-ticket can be used for several journeys.

Along this paper, the word 'journey' will usually refer to an 'elementary journey' in which a user travels from a starting to a destination station with no transfers.

Reusability is specially useful in cities providing several means of transport such as bus, tram, train, or underground. Citizens and tourists benefit from a great flexibility for planning their journeys inside the city.

The system must allow to check that an e-ticket has not expired, or that its maximum amount of elementary journeys has not been exceeded. In both cases, fraudulent uses must be detectable [17].

B. RELATED WORK

There exist plenty of proposals of e-ticket systems in the literature. Most of them do not consider the possibility to do transfers [10]–[12], [14], [16], [18]–[22], but implement mechanisms allowing clients to purchase and validate e-tickets anonymously. Specifically, the proposals [14], [19], [22] make use of pseudonyms so that the system can not link an e-ticket to the identity of the user who acquired it. Other proposals provide anonymity by means of alternative techniques [10]–[12], [18], [20], [21]. In case of fraudulent behavior, some proposals allow anonymity revocation [10]–[12], [14].

The proposal in [16] is the only one including the possibility of carrying out random inspections. An inspector can require a user to prove that she is the owner of the inspected e-ticket. At this moment the system is able to link the identity of the user to the e-ticket and get all the information about his journey (like the starting station).

There exist a few proposals allowing transfers so that the rider of a public transport vehicle can continue the trip on another bus or train [7], [15], [23]. In particular [7], [15] include revocable anonymity. They were both designed so as to be implemented on mobile devices. Some proposals limit the amount of transfers, whereas others allow an unlimited amount within a certain period of time.

Systems limiting the number of transfers usually implement a mechanism in which an e-ticket includes a hash chain whose size depends on the maximum number of allowed transfers. In this way, if $\mathtt{T}$ journeys are allowed, a $\mathtt{T}$-element hash chain $\{chain_k\}_{1\leq k\leq \mathtt{T}}$ is generated so that $chain_{k-1} = \mathcal{H}(chain_k)$, with $chain_{\mathtt{T}}$ being a random number. When beginning the i-th journey, the user is asked to provide $chain_i$.

To the best of our knowledge, no previous proposal allowing a limited amount of transfers includes the possibility of carrying out random inspections. In such a case, when a user identified herself and proved that she is the owner of the e-ticket, the system would be able to link her identity with all the journeys made with the inspected ticket. In our opinion, the system should only be able to determine whether the inspected user is authorized to make the current journey with the provided ticket.

C. CONTRIBUTION AND PLAN OF THIS PAPER

This paper proposes a construction to endow an existing e-ticket system with *reusability* so that a limited number of transfers can be done before an expiration time set at the moment of ticket validation.

Our proposal aims to be an extension, or *plug-in*, that can be coupled to any existing privacy-enabled ticketing system. After the purchase and validation of an e-ticket of the underlying system, the *plug-in* proceeds by generating a chain of *travel tokens*. These *travel tokens* are presented sequentially each time the user starts a new (elementary) journey at the beginning of her journey or after doing a transfer.

Regarding data privacy, the construction guarantees anonymity and unlinkability among all the journeys made with a given e-ticket, even after an inspection. This is the main contribution of the paper.

The proposal has been designed in order to be run on smart phones and get a *mobile application* similar to that used in the Iceland public transport system [24], but including privacy guarantees for users.

The paper has been structured in the following way. Section II introduces the cryptographic tools employed in the design of our solution. Section III explains the system and adversary models together with an enumeration of the privacy requirements. Section IV describes the new construction whereas Section V analyzes its privacy and security. Section VI shows the performance results obtained from a prototype implementation. Finally, Section VII concludes the paper.

II. PRELIMINARIES

The following section briefly reviews the cryptographic primitives and tools used in the design of our proposal. It also introduces the formal notation employed along the paper.

A. HASH AND HMAC FUNCTIONS

A *hash function*, $\mathcal{H}$, is a cryptographic one-way function that receives an arbitrary message, M, as input, and returns a fixed-length digest of M as output [25]. Such digests are recommended to be between 160 and 512 bit long.

It must be computationally hard to find two different messages, $M \neq M'$, such that $\mathcal{H}(M) = \mathcal{H}(M')$. Given a bitstring m, it must also be computationally hard to find a message M such that $m = \mathcal{H}(M)$.

A *keyed hash function*, *HMAC*, is a keyed cryptographic one-way function. Given a message M and a key K, we denote the resulting digest as $HMAC_K(M)$. The relation between a message M and its *HMAC* digest can only be determined if key K is known [26].

B. DIGITAL SIGNATURES

A digital signature is a cryptographic scheme which demonstrates the *authenticity* of a message M transferred through an insecure channel [27]. The scheme also guarantees *integrity* in the sense that the message has not been modified after being signed, and *non-repudiation*, *i.e.*, the signer can not deny having signed the message.

A signer, S, is required to own a private/public key pair. Then, $Sign_S(M)$ denotes the resulting signature on message M computed using signer's private key. Such a signature can be later validated using signer's public key. Signatures are usually computed over the hash digest of the message, $Sign_S(\mathcal{H}(M))$, so as to reduce its computational cost.

C. BLIND SIGNATURES

A blind signature [28] is computed by means of a protocol run between Alice, who is in possession of a message M, and Bob, who is in possession of a key pair. As a result, Alice obtains a signature by Bob on $\mathcal{H}(M)$ in such a way that Bob does not learn anything about $\mathcal{H}(M)$. Such a protocol consists of the following steps:

1) Alice selects a random *blinding factor*, r, and masks $\mathcal{H}(M)$ using r. The blinded result, $Blind_r(\mathcal{H}(M))$, is sent to Bob.
2) Bob signs the blinded message using his private key. The result is then returned to Alice.
3) Alice unblinds the received blinded signature obtaining a signature on $\mathcal{H}(M)$ by Bob, namely $Sign_{Bob}(\mathcal{H}(M))$.

The resulting signature can be validated by any party who is in possession of Bob's public key.

D. PARTIALLY BLIND SIGNATURES

Partially blind signature schemes [29] are an extension of blind signature schemes. They allow a signer, Bob, to explicitly include necessary information (expiration date, collateral conditions, or whatever) in the resulting signatures under some agreement with the message owner, Alice.

Bob is required to own a private-public key pair. Bob and Alice must agree with the information, *Info*, that will be included in the signature computed on Alice's message, M.

As a result of running a partially blind signature protocol, Alice obtains a digital signature by Bob on the digest of her message, $\mathcal{H}(M)$, together with the digest of the agreed information, $\mathcal{H}(Info)$. This signature will be denoted as $PartialSign_{Bob}(\mathcal{H}(M), \mathcal{H}(Info))$. Bob gets no information on $\mathcal{H}(M)$.

E. RANGE PROOFS OF COMMITTED VALUES

A *commitment scheme* [30] is a cryptographic primitive which allows one to commit to a chosen value while keeping it hidden to others, with the ability to reveal the committed value later. A commitment on E is computed from a random parameter c, so that $Commit_c(E)$ denotes the resulting commitment. The committed value is revealed by releasing the parameter c.

Some commitment schemes [31] allow to prove in zero-knowledge that a committed value lies in an interval $[a, b]$. Given $Commit_c(E)$, it is possible to prove in zero-knowledge that $a \leq E \leq b$.

III. SYSTEM AND ADVERSARY MODELS

This section first describes the system and adversary models. After that, the privacy requirements are detailed.

A. SYSTEM MODEL

Our system is composed of the following actors:

1) *Mobile application or app*. It runs on the mobile phone of public transport users. It is used for e-ticket purchase

and validation. Each time a user gets in a means of transport (after an e-ticket validation or a transfer), the mobile phone is to be approached to a hardware device which will grant access. When a user gets out, a similar operation is necessary for getting the data needed for the next journey in case a transfer is to be done. If a random inspection takes place, the user will use the *app* to demonstrate the validity of her ticket. The *app* also manages the so-called *deposit token*, employed to force all users to run the "get-out" procedure at the end of each journey, even when they are not going to do a transfer.

2) *System server.* This is a central server which manages all the information of the e-ticket system. It is contacted by the users' *app* during the purchase and validation of e-tickets. It is also accessed by transport system proximity devices when users run the "get-in" and "get-out" procedures, including the management of *deposit tokens*. Inspectors also access it when performing random inspections. It incorporates a database to store data and avoid double spending frauds.
3) *Transport system devices.* Our proposal requires the transport system provider to install a proximity reading device at each entrance and exit.

   - An *entrance device* is accessed by users when getting in a transport system. It is responsible for validating whether the data provided by a user (a *travel token*) allows her to get in the transport system. In that case, it grants access to the user and provides her with data (an *inspection token*) that will be requested if an inspection takes place.
   - An *exit device* is accessed by users when leaving a transport system. This access is mandatory for all the users even if they are not going to do a transfer. If not done, the user loses her *deposit token*.

   Both devices need a permanent communication with the system server.
4) *Inspector.* He is responsible for carrying out random controls within the transport system. He verifies that users are traveling with a valid *inspection token*. The inspector uses a mobile device to that end.

The use cases are next summarized:

1) *Server set-up.* The system server is initialized. After configuring the underlying e-ticket system, the *plug-in* is also configured by setting some parameters like the maximum amount of allowed transfers or the expiration time. Several cryptographic keys are also created at this moment.
2) *Application set-up.* Public transport users install and configure the *app* in their mobile phones. Some personal data is requested at this stage.
3) *E-ticket purchase.* A user acquires, via *app*, e-tickets of the underlying e-ticket system by following the established procedure.

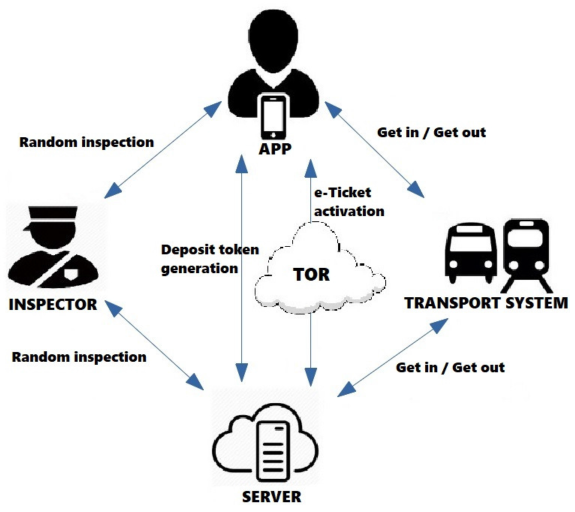

FIGURE 1. System model.

4) *Deposit token generation.* So as to force users to run the "get-out" procedure at the end of each journey, they leave a *deposit token* during the "get-in" process, and get a new (free of charge) one when they do "get-out". If not done, the user loses her *deposit token* so that she will have to pay for a new one. This procedure serves to that end.
5) *E-ticket activation.* The *app* takes an e-ticket of the underlying system and connects to the server so as to activate it. After that, the *plug-in* generates a chain of *travel tokens*. The user can now begin her journey by performing a "get-in" action using the first *travel token* of the chain. So as to avoid tracing, this procedure should be run through an anonymous channel [32].
6) *Get-in.* When the user is to begin a new journey (after activating an e-ticket or after doing a transfer), she is required to approach her mobile phone to a proximity device located at the entrance control of the transport system. After the user provides a *ready to be used travel token* and a valid *deposit token*, she will receive an *inspection token* and will be granted access to the transport system. Otherwise, the system will deny access by means of a blocking barrier or a warning sound.
7) *Get-out.* When leaving a transport system, a user is required to approach her phone to a proximity device similar to that accessed during the "get-in" procedure. In this way, her next *travel token* becomes *ready to be used*, and she obtains a new free of charge *deposit token*.
8) *Random inspection.* An inspector requires a user to prove she is allowed to make the current journey. The user has to approach her mobile phone to inspector's and provide her *inspection token*. In case of infringement the user will be fined.

B. ADVERSARY MODEL

We assume that all the cryptographic primitives have been set so that an adversary can not perform successful brute-force attacks against them. Regarding the capacity of an attacker:

1) An adversary can not corrupt the *app*. The *app* does not leak any private information about users. It runs all the protocols as specified.
2) The system server and the inspectors are honest–but–curious entities. They run the protocols as specified, but may collaborate with an adversary by releasing all the information they have access to.

From the service provider point of view, users are untrusted entities who may try to travel for free.

C. DESIGN OBJECTIVES

The privacy and security goals to be achieved by our proposal are enumerated next:

1) While traveling, the interactions with the system (except for those arising from an inspection carried out by an inspector) performed by a user are anonymous and unlinkable.
2) While traveling, a user is only required to identify herself if requested by an inspector. In such a case, the only information obtained by the system is a *boolean* indicating whether the identified user is allowed to make the current journey.
3) Users traveling without a valid *inspection token* will be detected if requested by an inspector. An *inspection token* is valid when it is linked to user's identity, and has been issued for the current public transport trip.

Requirement 1 implies that the moments a user gets in and out of a transport system can not be related. Also, the system can not link the different journeys made by a user with one or several e-tickets.

Requirement 2 determines that, as a result of an inspection, the system can only obtain information inherent to the inspection itself. That is, it can associate a user with a specific time and place. Any additional information, like the starting and end stations can not be deduced.

Finally, Requirement 3 states that users can not travel fraudulently. It is worth mentioning that an electronic e-ticket system can not provide security against certain actions like users getting in a transport system after jumping over the access blocking barrier. Such users will be fined if requested by an inspector.

IV. OUR PROPOSAL

A. SYSTEM OVERVIEW

Our proposal is designed as a *plug-in* to be coupled to an existing e-ticket system. The underlying e-ticket system has its own purchase and anonymity policies.

When a user starts a new journey (maybe composed of several elementary journeys with the corresponding transfers), she must activate an available e-ticket of the underlying system. Just after that, the ''travel token chain generation'' procedure is run. During its execution, the *plug-in* generates a *travel token* chain whose length equals the maximum amount of elementary journeys allowed per e-ticket. When this procedure concludes, the first *travel token* of the chain is *ready to be used*. These chained *travel tokens* are to be presented sequentially at the beginning of each elementary journey.

Just before getting access to a transport system vehicle or platform, the user performs a ''get-in'' operation during which she is required to approximate her mobile phone to an entrance control device and provide a *travel token* via some proximity communication system. The entrance device, by querying the central system server, determines whether the provided *travel token* is *ready to be used* and checks that it has not been used before. If the validation is correct, the entrance device will generate, and send to the mobile device, an *inspection token* which includes a *trip identifier*. A *trip identifier* is a random value which uniquely identifies each trip carried out by the transport system. The *inspection token* will be required in case of inspection.

When a user leaves a platform or vehicle, she runs the ''get-out'' procedure. She must then approximate her mobile phone to an exit device in order to get her next *travel token ready to be used* for the next elementary journey.

During a ''random inspection'', the user must provide the *inspection token* she obtained when running the ''get-in'' procedure. That *inspection token* is linked to some piece of personal information, such as her passport number or the hash digest of a personal picture, which will be checked by the inspector.

So as to avoid some fraudulent uses, our system requires all the users to run the ''get-out'' procedure at the end of each elementary journey even when they are not going to do a transfer. That is achieved by making users provide a *deposit token* during the execution of the ''get-in'' procedure and obtaining a free of charge new one when running the corresponding ''get-out''. If that is not done, the user loses her deposit and will have to pay for a new one. The first *deposit token* of each user is provided free of charge.

B. PRELIMINARY SETUP PROCEDURES

Next we describe two procedures related to system configuration. The first one, ''Server set-up'', is run by the ticket issuer so as to create and set the cryptographic keys and constants required for the system to run. The other one, ''Application set-up'', is run by users when installing the system *app* in their mobile devices.

1) SERVER SET-UP

Before deploying the system, the ticket issuer:

1) Creates the following cryptographic keys:
   - ''Get-in'' blind signature (see Sect. II-C) key pair: $P_{S^{in}}/V_{S^{in}}$.
   - ''Get-out'' blind signature key pair: $P_{S^{out}}/V_{S^{out}}$.
   - ''Deposit'' blind signature key pair: $P_{S^{d}}/V_{S^{d}}$.
   - ''User token'' blind signature key pair: $P_{S^{u}}/V_{S^{u}}$.

- "Trip" partially blind signature (see Sect. II-D) key pair: P_{S^t}/V_{S^t}.

  The created public keys are: $P_{S^{in}}, P_{S^{out}}, P_{S^d}, P_{S^u}, P_{S^t}$.
  The created private keys are: $V_{S^{in}}, V_{S^{out}}, V_{S^d}, V_{S^u}, V_{S^t}$.

2) Creates a tuple of setup parameters for a commitment-scheme allowing zero-knowledge range proofs (see Sect. II-E).
3) Sets and stores the following constants:
   - $\mathtt{T} \equiv$ Maximum amount of journeys allowed per e-ticket.
   - $\mathtt{TIME\_ACTIVE} \equiv$ Amount of minutes during which an e-ticket is valid after its activation.
   - $\mathtt{TT}_{\mathtt{end}} \equiv$ Random string.
   - $\mathtt{N} \equiv$ Parameter which tunes the security level of the cut-and-choose checking performed during the generation of *travel token* chains.

The private keys $V_{S^{in}}, V_{S^{out}}, V_{S^d}, V_{S^u}, V_{S^t}$ must be kept secret by the ticket issuer which will make them available only to the system server. The remaining parameters are made public to all the actors.

2) APPLICATION SET-UP

A user has to install the application of the underlying e-ticket system together with the extension *plug-in* on her mobile phone. Then, she must introduce all the information required by the underlying e-ticket system (credit card, etc). After that, the extension *plug-in* will request some information that identifies her unequivocally (like a picture or her passport number) which will be stored in the ID_{user} object in her phone. This information must allow an inspector to verify her identity.

The *app* obtains a free of charge *deposit token* by running the procedure described in Section IV-D1.

C. TRAVEL TOKEN *CHAIN*

After a ticket is activated, the system generates a chain composed of $\mathtt{T}$ chained *travel tokens*, $\{TT_i\}_{0\le i<\mathtt{T}}$, being $\mathtt{T}$ the maximum amount of journeys (with the corresponding transfers) that can be made with each e-ticket. Each *travel token* in a given chain is assigned the same expiration time E and is linked to the same masked user identity ID_{masked} (computed as $HMAC_{K_u}(ID_{user})$ with K_u being a secret key).

1) STRUCTURE OF A TRAVEL TOKEN

A *travel token* is a tuple containing several public parameters computed from a set of secret ones. The private parameters are generated and kept secret by the *app*. The public ones compose the *travel token* object, TT_i.

- Private parameters of TT_i:
  - K_i: secret key used to re-mask the masked user identity.
  - u_i: secret value for blinding the re-masked user identity.
  - c_i: secret value for committing to the expiration time.
  - t_i: secret value for blinding the link to the next *travel token* of the chain.
- Public parameters of TT_i:
  - U_i: blinded re-masked user identity, namely $Blind_{u_i}(\mathcal{H}(HMAC_{K_i}(ID_{masked})), P_{S^u})$.
  - C_i: commitment to the expiration time E, namely $Commit_{c_i}(E)$.
  - $NextTT_i$: blinded link to the next *travel token* of the chain, namely $Blind_{t_i}(\mathcal{H}(TT_{i+1}), P_{S^{in}})$.
- The chain is concluded by setting $TT_{\mathtt{T}} = \mathtt{TT}_{\mathtt{end}}$.

A *travel token* TT_i is said to be *ready for use* when it is accompanied by two digital signatures under the "get-in" and "get-out" server key pairs, namely $Sign_{S^{in}}(\mathcal{H}(TT_i))$ and $Sign_{S^{out}}(\mathcal{H}(TT_i))$. A *ready for use travel token* allows a user to get access to a transport system.

2) GENERATION OF A *TRAVEL TOKEN* CHAIN

A *travel token* chain is generated by means of the following procedure which takes as input and expiration time E and a masked user identity ID_{masked}:

1) Set $i = \mathtt{T} - 1$
2) While $i \geq 0$ (generate TT_i):
   - Generate K_i, u_i, c_i, and t_i at random and store them secretly.
   - Set $U_i = Blind_{u_i}(\mathcal{H}(HMAC_{K_i}(ID_{masked})), P_{S^u})$.
   - Compute $C_i = Commit_{c_i}(E)$.
   - Compute $NextTT_i$:
     - if $i < \mathtt{T} - 1$ then
       $NextTT_i = Blind_{t_i}(\mathcal{H}(TT_{i+1}), P_{S^{in}})$.
     - if $i = \mathtt{T} - 1$ then
       $NextTT_i = Blind_{t_i}(\mathcal{H}(\mathtt{TT}_{\mathtt{end}}), P_{S^{in}})$.
   - Pack U_i, C_i and $NextTT_i$ into a tuple and name it TT_i.
   - Let $i = i - 1$.
3) The resulting *travel token* chain is represented as a set $\mathcal{C} = \{TT_0, \ldots, TT_{\mathtt{T}-1}\}$.

3) VALIDATION OF A *TRAVEL TOKEN* CHAIN

Given $\mathcal{H}(TT_0)$, the correct composition of a *travel token* chain can be checked by requiring its generator to provide the input parameters E and ID_{masked} together with all its private and public parameters. The verifier can then repeat the construction process from $TT_{\mathtt{T}-1}$ down to TT_0 and finally check that the hash of the obtained token TT_0 matches $\mathcal{H}(TT_0)$.

D. TRAVEL-RELATED PROCEDURES

The following section describes the protocols executed by users while traveling in public transport.

1) DEPOSIT TOKEN GENERATION

When a user gets in a transport system, she has to provide a *deposit token*. Later, when she gets out, she receives a new one which is generated by means of the process described in this section. As a result of this process, the user gets a new *deposit token* whose structure is that of an anonymous e-coin generated through Chaum's system [28].

A *deposit token* is generated free of charge the first time a user registers to the system or when obtained during the "get-out" process. If a user lost her *deposit token*, probably because she did not run the "get-out" procedure at the end of a journey, she will be charged for obtaining a new one.

A *deposit token* is generated as follows:

1) If necessary, the *app* performs a credit card payment for the cost of a *deposit token*.
2) The *app* generates a random value *deposit*, hashes and blinds it with a random blinding factor d for server's deposit key, $Blind_d(\mathcal{H}(deposit), P_{\mathcal{S}^d})$, and sends the resulting value to the server.
3) The server blindly signs the received value and returns the resulting blinded signature to the *app*.
4) The *app* unblinds the received blinded signature getting a signature by the server on $\mathcal{H}(deposit)$, namely $Sign_{\mathcal{S}^d}(\mathcal{H}(deposit))$.
5) Finally, the *app* stores the resulting *deposit token* tuple $\{deposit, Sign_{\mathcal{S}^d}(\mathcal{H}(deposit))\}$.

When the previous procedure is executed during a "get-out" process (step 1 is skipped), the identity of the user is not revealed.

When a payment is done, anonymity is not required since the identity of the user can be obtained from the payment method data. Such use case shall not be applied to users acting as required.

2) E-TICKET PURCHASE

The user pays for and gets e-tickets of the underlying e-ticket system. The underlying e-ticket system is required to guarantee that the purchased e-tickets can later be activated anonymously.

3) E-TICKET ACTIVATION

When a user is about to start a journey (which may include several transfers), her *app* contacts the ticket issuer server and activates an e-ticket of the underlying system. After a successful activation, the *plug-in* module runs the following "travel token chain generation" procedure.

Travel token chain generation. This procedure generates a chain composed of $\mathtt{T}$ *travel tokens*. Its correct composition is checked by means of a cut-and-choose technique tuned with parameter $\mathtt{N}$.

1) The *app* sets the expiration time to $E = CurrentTime + \mathtt{TIME\_ACTIVE}$.
2) The *app* takes ID_{user} and computes a set of $\mathtt{N}$ masked identities $\{ID_{masked_k} = HMAC_{K_{U_k}}(ID_{user})\}_{0\leq k<\mathtt{N}}$ from $\mathtt{N}$ different keys K_{U_k}.
3) Using the procedure of Section IV-C2, the *app* generates $\mathtt{N}$ *travel token* chains $\{\mathcal{C}^k\}_{0\leq k<\mathtt{N}}$ providing E and ID_{masked_k} as input, respectively.
4) The app, for each chain $\mathcal{C}^k = \{TT_0^k, \ldots, TT_{\mathtt{T}-1}^k\}$, hashes its first *travel token*, TT_0^k, and blinds the resulting digest under a random blinding factor r_k generating a set of blinded digests:
$$\{B_k = Blind_{r_k}(\mathcal{H}(TT_0^k), P_{\mathcal{S}^{in}})\}_{0\leq k<\mathtt{N}}.$$
All these blinded values are sent to the server.
5) The server chooses a random $j \in [0, \mathtt{N}-1]$ and sends it to the *app*.
6) The app, for each $k \neq j$, sends the blinding factor r_k to the server together with all the information required to check the correct composition of chain $\mathcal{C}^k$.
7) For each $k \neq j$, the server unblinds the received hash and verifies each chain C^k under $\mathcal{H}(TT_0^k)$, E and ID_{masked_k} together with all the additional parameters as described in Section IV-C3.
8) If all the validations are satisfied, the server blindly signs B_j under its "get-in" key pair so that the *app* is able to obtain $Sign_{\mathcal{S}^{in}}(\mathcal{H}(TT_0^j))$.
9) Next, the *app* generates r'_j at random and computes $Blind_{r'_j}(\mathcal{H}(TT_0^j), P_{\mathcal{S}^{out}})$ which is sent to the server which will blindly sign it under its "get-out" key pair. From the result, the *app* obtains the signature $Sign_{\mathcal{S}^{out}}(\mathcal{H}(TT_0^j))$.
10) Finally, the *app* keeps a chain of *travel tokens* $\mathcal{C}^j = \{TT_0^j, \ldots, TT_{\mathtt{T}-1}^j\}$, whose first token, TT_0^j has been blindly signed by the server under the "get-in" and "get-out" key pairs. Hence, this first *travel token* is *ready for use*.

If some checking fails, the procedure of the underlying e-ticket system applied in case of fraudulent behaviour is run. This may include anonymity revocation and fining the user. Although the probability of cheating can not be made arbitrarily small (making $1/N$ negligible would require a rather large value for N which has a direct impact on the running time of the protocol), users will be discouraged from cheating if the amount of the fine is large enough.

The chain generation process ensures that all the *travel tokens* of a chain are linked to the same ID_{user} object so that they can not be shared among different users. The user linked to the *travel tokens* of the chain shall not necessarily be the one who acquired the e-tickets. This assumes that the underlying e-tickets are transferable.

4) GET-IN

When a user is about to begin a new journey (after doing a transfer or after the activation of an e-ticket), it approaches her mobile phone to a device located at the entrance control system. Both devices communicate through some proximity communication system. If the user provides a *ready to be used travel token* (signed under both the "get-in" and "get-out" key pairs) and a valid *deposit token*, she will be allowed to enter. The transport system, for each trip, generates a unique identifier ID_{trip}. The user receives an *inspection token* that will be required in case of inspection.

The user is required to leave a *deposit token*. In this way, she will be forced to run the "get-out" process at the end of her journey so as to get a new one. If not done, she will lose it and will have to pay for a new one.

1) The *app* transmits its *deposit token*, namely $\{deposit, Sign_{S^d}(\mathcal{H}(deposit))\}$.
2) The entrance device verifies the digital signature over $\mathcal{H}(deposit)$ and asks the system server to check it has not been used before. If all these checkings are satisfied, the server stores $\mathcal{H}(deposit)$ in a database to record it as already used.
3) The *app* transmits a *ready to be used travel token* TT_i (accompanied with $Sign_{S^{in}}(\mathcal{H}(TT_i))$ and $Sign_{S^{out}}(\mathcal{H}(TT_i))$) to the entrance device.
4) The entrance device verifies both digital signatures and checks that $TT_i \neq \mathtt{TT_{end}}$.
5) The entrance device computes the digest $\mathcal{H}(TT_i)$ and sends it to the system server to verify that it has not been used before. If not used before, the server stores $\mathcal{H}(TT_i)$ together with *CurrentTime* to record it as already used.
6) The *app* proves in zero-knowledge to the entrance device that the commitment C_i extracted from TT_i contains an expiration time that is still valid by proving that it falls between *CurrentTime* and $CurrentTime + \mathtt{TIME\_ACTIVE}$. This is done by means of the techniques described in Section II-E.
7) The entrance device extracts $NextTT_i$ from TT_i and asks the server to sign it with the "get-in" key, so that the *app* blindly obtains a signature $Sign_{S^{in}}(\mathcal{H}(TT_{i+1}))$.
8) The entrance device extracts U_i from TT_i and asks the server to sign it with the user-token key, so that the *app* blindly obtains $U_{token} = Sign_{S^u}(\mathcal{H}(HMAC_{K_i}(ID_{masked})))$.
9) The user asks, via the entrance device, the server to compute a partially blind signature on U_{token} with ID_{trip} as agreed information. The resulting signature $U_{trip} = PartialSign_{S^t}(\mathcal{H}(U_{token}), \mathcal{H}(ID_{trip}))$ is obtained by the *app*.
10) Finally, the *inspection token* is stored by the *app* as the tuple $\{U_{token}, ID_{trip}, U_{trip}\}$.

5) GET-OUT

When a user has finished a journey and is about to leave the transport system, she has to approach her mobile phone to the corresponding exit device. As a result of this protocol, the user gets her next *travel token ready to be used* (she receives a second signature under the "get-out" key pair) and a new free of charge *deposit token* to be used in her next journey. She is also asked to present her *inspection token* so as to record it as no longer valid.

If this protocol is not run, the user does not get a *deposit token* and will have to pay for a new one so as to use the public transport again in the future.

1) The *app* hashes and blinds, with a random number t, the next *travel token* TT_{i+1}. Then it sends the result, $Blind_t(\mathcal{H}(TT_{i+1}), P_{S^{out}})$, to the exit device.
2) The exit device asks the system server to sign the received blinded message under the "get-out" key pair so that the *app* obtains $Sign_{S^{out}}(\mathcal{H}(TT_{i+1}))$.
3) The *app* obtains a free *deposit token* as follows:
   a) The *app* sends the current *inspection token* $\{U_{token}, ID_{trip}, U_{trip}\}$ to the exit device together with $HMAC_{K_i}(ID_{masked})$.
   b) The exit device, checks ID_{trip} corresponds to the current trip of the transport system.
   c) The exit device, in collaboration with the system server, verifies that neither U_{token} nor U_{trip} have been received before. It also verifies that U_{token} is a valid signature on $\mathcal{H}(HMAC_{K_i}(ID_{masked}))$. It finally verifies that U_{trip} is a partial signature $PartialSign_{S^t}(\mathcal{H}(U_{token}), \mathcal{H}(ID_{trip}))$.
   d) By running the procedure in Section IV-D1, the *app* gets a new free of charge deposit token, $\{deposit, Sign_{S^d}(\mathcal{H}(deposit))\}$.
   e) Finally, the system server stores both U_{token} and U_{trip} so as to record them as no longer valid.

6) RANDOM INSPECTION

During an inspection, an inspector determines whether a user has a valid *inspection token* for the current public transport trip.

1) The *app* sends $\{ID_{user}, ID_{trip}, U_{token}, U_{trip}, K_u, K\}$ to the inspector via a short-range communication system.
2) The inspector:
   a) Checks that the data in ID_{user} corresponds to the person in from of him (it may be a picture or her passport number).
   b) Checks that ID_{trip} corresponds to the current trip.
   c) Checks that neither U_{token} nor U_{trip} have been provided in a previous inspection nor have been marked as no longer valid.
   d) Computes $ID_{masked} = HMAC_{K_u}(ID_{user})$, and checks that U_{token} is a valid signature $Sign_{S^u}(\mathcal{H}(HMAC_K(ID_{masked})))$.
   e) Checks that U_{trip} is a valid partial signature $PartialSign_{S^t}(\mathcal{H}(U_{token}), \mathcal{H}(ID_{trip}))$.
3) The inspector sends both U_{token} and U_{trip} to the server so as to record them as no long valid.

If some of these checkings fails, the user will be fined according to the established regulations.

After being inspected, the user obtains a new *inspection token* so as to avoid that her next execution of the "get-out" process can be linked to the inspection. This is done as follows:

1) The *app* generates a random key K', a random blinding factor u', and computes $U' = Blind_{u'}(\mathcal{H}(HMAC_{K'}(ID_{masked})), P_{S^u})$ which is sent, via inspector's device, to the server.
2) The server blindly signs U' and returns, via inspector's device, the result to the *app* so that it can obtain $U'_{token} = Sign_{S^u}(\mathcal{H}(HMAC_{K'}(ID_{masked})))$.

3) Next, the *app* asks, via inspector's device, the server to compute a partially blind signature over U'_{token} with ID_{trip} as agreed information. The resulting signature $U'_{trip} = PartialSign_{S^t}(U'_{token}, ID_{trip})$ is obtained by the *app*.
4) Finally, the *app* stores the new *inspection token* as the tuple $\{U'_{token}, ID_{trip}, U'_{trip}\}$.

V. PRIVACY AND SECURITY ANALYSIS

This section analyzes the security of our proposal under the assumed adversary model (Sec. III-B), and the privacy and security requirements described in Section III-C.

A. PRIVACY ANALYSIS

We next present a set of lemmas which are the basis for proving Theorem 1.

Lemma 1: During an execution of the "travel token chain generation" procedure, the server obtains no information allowing it to identify the generated chain.

Proof: A *travel token* chain is generated through a cut-and-choose protocol in which the *app* first generates $\mathtt{N}$ blinded chains. After that, the server asks the *app* to reveal all the parameters used in the generation of $\mathtt{N}-1$ of them. Finally, the first *travel token* of the remaining chain is blindly signed by the server. We next justify the server gets no information allowing it to identify the remaining chain.

All the random parameters generated during the creation of the $\mathtt{N}$ chains are chosen independently among them so that revealing the parameters of $\mathtt{N} - 1$ chains does not provide any information about the parameters used for creating the remaining one.

Regarding the ID_{user} object, it is masked with a different random key before its inclusion in each chain (set $\{ID_{masked_k}\}_{0 \leq k < \mathtt{N}}$). These masked values are revealed for $\mathtt{N} - 1$ of the chains, but their knowledge does not provide any information about the remaining one, since the random keys are kept secret.

The $\mathtt{N}$ chains share the same expiration time E which is revealed during the cut-and-choose checkings. Nevertheless, E is included in the travel tokens TT_i through commitments $C_i = Commit_{c_i}(E)$. In all future uses of these tokens, the non-expiration of a *travel token* is proven by means of zero-knowledge proofs so that the exact value for E is never revealed. Hence, the remaining chain can not be identified from E.

Finally, the only access the server has to the remaining chain is produced when it is asked to blindly sign its first *travel token*. A blind signature protocol guarantees that the signing party gets no information about the signed data. Hence, the server gets no information about the signed token.

Lemma 2: The *travel tokens* of a chain can not be linked among them by the server.

Proof: A *travel token* TT_i is a tuple composed of three parameters, namely U_i, C_i, and $NextTT_i$.

Both U_i and $NextTT_i$ are blinded values generated from random blinding factors so that they can not be related to the values blinded inside them nor to the resulting signatures. Hence, they provide no information at all as long as the blinding factors are kept secret by the *app*.

The expiration time, E, committed in C_i is shared among all the *travel tokens* of a chain. However, each C_i is generated from a different and random c_i so that linking *travel tokens* from these commitments is computationally unfeasible. The non-expiration of a *travel token* is proven through zero-knowledge proofs so that the value E is not revealed.

Lemma 3: When an *inspection token* is generated, the server only gets information about its ID_{trip} component.

Proof: An inspection token is a tuple $\{U_{token}, ID_{trip}, U_{trip}\}$. The U_{token} component is obtained by the *app* as a blind signature computed by the system server. Hence, the server obtains no information about it.

Regarding U_{trip}, it is obtained by the *app* as a partially blind signature over U_{token} with ID_{trip} as agreed information. Hence, the server only learns ID_{trip}.

Lemma 4: After a "random trip inspection", the "get-in" and "get-out" procedures of the inspected journey can not be determined.

Proof: During an inspection, the *app* sends the tuple $\{ID_{user}, U_{token}, ID_{trip}, U_{trip}, K_u, K\}$ to the inspector.

The link between an *inspection token* and the *travel token* TT_i provided for its generation is given by U_{token} being a signature over the value blinded in the U_i component of TT_i. Since U_i is a blinded value, it can not be related to the (unblinded) signature obtained from it. Hence, an *inspection token* provided during an inspection can not be related to the execution of the "get-in" process it was generated in.

At the end of an inspection, a new *inspection token* is generated by the user. When it will be provided during an execution of the "get-out" procedure, it will not be linkable to the current inspection.

Theorem 1: The proposed system meets the design objectives 1 and 2 (user's mobility profiles can not be created).

Proof: When a user activates an e-ticket of the underlying system, a chain of *travel tokens* is created. From Lemma 1, the server gets no information allowing it to identify that chain, hence the "e-ticket activation" operation keeps anonymous and unlinkable.

After that, each time the user runs a "get-in" procedure, she provides a *travel token* of that chain. From Lemma 2, the tokens of a chain can not be related among them so that the executions of the "get-in" process using tokens of the same chain can not be related among them.

When running a "get-in" process, the user also obtains an *inspection token*, from which, according to Lemma 3, the server gets no information but its ID_{trip} component. Hence the server gets no information at all about the user from the "get-in" process.

When running a "get-out" process, the user provides her *inspection token* so as to mark it as no longer valid. Since the keys used to mask user's identity in that token are not revealed, the server can not link that token to the identity of the user. From Lemma 3, the provided *inspection token* can not be linked to any personal data of the user.

When a random inspection takes place, the user is asked to provide her *inspection token* and all the data allowing to link it to her identity. The inspector gets user's identity but, according to Lemma 4, her *inspection token* can not be linked to the executions of the "get-in" and "get-out" procedures of that journey. In this way, the only information obtained is the presence of the inspected user at the place where the inspection has taken place.

B. SECURITY ANALYSIS AGAINST MALICIOUS USERS

We next present a set of lemmas which are the basis for proving Theorem 2.

Lemma 5: A malicious user can get a fraudulent *travel token* chain with probability at most $\frac{1}{N}$.

Proof: A *travel token* chain has been properly generated when its first *travel token*, TT_0, is *ready to be used*, *i.e.*, it has been digitally signed by the system server. Since digital signatures are unforgeable, the only way for a user to get such signatures is to run the "travel token chain generation" procedure, in which the server signs the first token of a chain.

A malicious user could cheat by trying to get a chain with a larger amount of *travel tokens*, with a higher expiration time, or with its *travel tokens* linked to different identity objects.

A *travel token* chain is generated using a cut-and-choose technique in which the user generates $\mathtt{N}$ chains. Next the server asks her to reveal all the parameters used in the generation of $\mathtt{N} - 1$ of them so that it can check their correct composition. Finally, it signs the remaining unrevealed one. So as to succeed, a malicious user must generate just one malicious chain and be lucky so that the fraudulent chain is the one which is not checked by the server. This happens with probability $\frac{1}{N}$.

Lemma 6: A user can not obtain a fraudulent *ready to be used travel token* from a properly generated *travel token* chain.

Proof: A *ready to be used travel token* is a non-expired *travel token*, TT_i, accompanied with two digital signatures computed under the "get-in" and "get-out" server key pairs ($Sign_{S^{in}}(\mathcal{H}(TT_i))$ and $Sign_{S^{out}}(\mathcal{H}(TT_i))$). Since digital signatures are unforgeable, the only way for a user to get such signatures is by requesting them to the server.

The first *travel token* of a chain is signed during the chain generation process. If we assume that the chain has been properly generated, its first *travel token* can not be fraudulent.

The signatures over the following *travel tokens* are obtained through the "get-in" and "get-out" procedures. During the execution of the "get-in" procedure, the server blindly signs the next *travel token* by signing, under its "get-in" key pair, the $NextTT_i$ field of a *ready for use travel token* provided by a user. Hence, the next *travel token* can only be fraudulent if the provided one also is. That is not possible if we assume the chain was properly generated.

During the "get-out" process, the server blindly signs under its "get-out" key pair without performing any checking. Nevertheless, a signature under the "get-out" key pair is useless if the corresponding one computed with the "get-in" key pair is not available. That signature under the "get-out" key pair is computed just to complicate the transmission of *ready for use travel tokens* among users.

Lemma 7: An *inspection token* is linked to the ID_{user} object contained in the *travel token* provided during its generation, or some user lost a deposit token.

Proof: An *inspection token* is generated during the execution of the "get-in" procedure. A user provides a *deposit token* and a *ready to be used travel token*, TT_i, to the server which extracts its U_i field and computes a digital signature over it so that the user obtains a blind signature $U_{token} = Sign_{S^u}(\mathcal{H}(HMAC_{K_i}(ID_{masked})))$ linked to ID_{masked}, which is linked to ID_{user}. After that, the user requires the server to compute a partially blind signature on U_{token} with ID_{trip} as agreed information. The resulting signature obtained by the user is denoted U_{trip}. The resulting *inspection token* is the tuple $\{U_{token}, ID_{trip}, U_{trip}\}$. Hence, the obtained *inspection token* is linked to ID_{user} through its U_{token} component.

During the "get-out" procedure, the user will be required to provide her *inspection token* so as to get a free of charge *deposit token*. All the components of that *inspection token* will be recorded as no longer valid.

A malicious user may perform the "get-in" procedure but require a partially blind signature U'_{trip} over a value U'_{token} (linked to a different ID'_{user}) different from that obtained from the provided TT_i. Since U'_{token} is a digital signature, it can not be forged, so that it must have been obtained during a "get-in" procedure whose corresponding "get-out" has not been performed. In that case, the *deposit token* was not obtained, so that the user lost it.

Theorem 2: The proposed system meets the design objective 3 (a user can not travel fraudulently).

Proof: Lemma 5 proves that, for a properly chosen parameter $\mathtt{N}$, a *travel token* chain is probably generated with the allowed amount of *travel tokens*, with a valid expiration time, and with all its *travel tokens* linked to the same user identity.

Next, Lemma 6 ensures that a *travel token* accepted during a "get-in" procedure can not be fraudulent. Additionally, the system stores the used *travel tokens* so as to avoid them being used more than one time.

Finally, Lemma 7 guarantees that the *inspection token* generated during a "get-in" procedure is linked to the identity of the provided *travel token*, or that, in case of fraud, the malicious user lost a *deposit token*. If the price of a *deposit token* is set to be higher than that of regular e-tickets, users are discouraged from acting fraudulently.

VI. EXPERIMENTAL RESULTS

The actors considered in our experiments are: the *server*, the *entrance devices*, the *exit devices*, and the *app*. To simplify, we have considered the *entrance* and *exit devices* to be part of the *system server*.

Prototypes have been implemented in `Java`. The *app* has been developed specifically for the `Android` operating system. The `java.math.BigInteger` library has been used to implement cryptographic operations involving big integers.

The feasibility of the system has been analyzed from the two most critical use cases. Namely, "e-ticket activation" (Section IV-D3) which performs a cut-and-choose checking involving strong computations, and the "get-in" process (Section IV-D4) which should respond with a minimum delay. Both protocols are executed between the *app* and the *system server*.

Our implementation makes use of RSA blind signatures [28], Abe and Okamoto's partially blind signatures [29] and Boudot's range proofs for committed values [31]. All these protocols have been chosen because no successful cryptanalysis attacks against them have been reported in the last years. A deep research would probably provide alternative options leading to faster running times. Nevertheless, the objective of this section is just to show the feasibility of the proposed system. Optimizing its performance falls beyond our objectives.

Our experiments employ 2048 bit cryptography for RSA blind signatures and for all the operations involving hard-to-solve instances of the discrete logarithm problem. Hash digests are computed by means of the SHA-256 function.

The cut-and-choose checking run during "e-ticket activation" has been tuned to $\mathtt{N} = 100$ so that the probability of cheating is reduced to 1% (generation of a fraudulent chain involves, on average, the payment of 99 fines). The "travel token chain generation" procedure has been executed with parameter $\mathtt{T} = 3$ (maximum amount of elementary journeys which can be done with a single e-ticket). All the implemented processes require intensive CPU usage with negligible memory requirements.

TABLE 2. Mobile application running times (in milliseconds).

Mobile phone			**E-ticket activation**		**Get-in**
BQ Model	Cores	GHz	Real-time	Pre-computed	Serial
Aquaris U Lite	4	1.4	21528	8952	880
Aquaris U	8	1.4	16414	4467	877

Table 2 shows the average running times at the *app*. Two variants of the "e-ticket activation" protocol have been tested. A *real-time* variant in which all the computations are performed during the protocol execution, and a *pre-computed* variant in which values for ID_{masked}, U_i, and $NextTT_{\mathtt{T}-1}$ have been computed in advance. Both variants have been implemented taking advantage of parallel computing. We can observe that pre-computations provide a 60%-75% time reduction. The "get-in" protocol at the *app* has been implemented in serial mode (it can not benefit neither from pre-computations nor parallel processing). Running the "get-in" protocol took less than one second in the two devices it was tested on.

TABLE 3. System server running times (in milliseconds).

System server				**E-ticket activation**		**Get-in**	
Processor	Cores	Threads	GHz	Serial	Parallel	Serial	Parallel
Intel i5-4310	2	4	3.00	17163	7719	320	148
AMD Athlon	4	4	2.80	25867	7497	487	126
Intel i7-6700	4	8	3.40	11812	2814	219	56
Intel i7-8700	6	12	3.20	10103	1615	184	30

Table 3 shows the average running times at the server part. Both the "e-ticket activation" and "get-in" protocols have been executed in serial and in parallel. In the parallel version, multiple instances of the "e-ticket activation" protocol have been executed concurrently.

In our opinion, the experiments show that the *app* can be deployed on current mobile phones, since tickets can be activated in around 4.5 seconds, whereas a "get-in" procedure can be completed in less than one second. The computer load at the server part must be balanced among several computers so as to manage all the concurrent requests received in a large city. A professional 36-Cores (72-Threads) computer could manage three "e-ticket activation" requests per second.

VII. CONCLUSION

This article has presented a construction for providing reusability to mobile phone-based e-tickets that avoids the possibility to create profiles about user's mobility habits. Reusability is provided in the sense that users, after activating an e-ticket of the underlying system, can make several journeys before an expiration time. The maximum amount of journeys is determined from the length of a chain of *travel tokens* that is generated after the activation of an e-ticket.

The proposal incorporates the possibility to carry out random *in situ* inspections. The experiments have shown that the proposal could be run on nowadays mobile phone devices.

REFERENCES

[1] *First Organized Public Transit System*. Accessed: May 26, 2020. [Online]. Available: https://www.wired.com/2008/03/march-18-1662-the-bus-starts-here-in-paris

[2] T. E. W. Khattak and P. Buhler, *Big Data Fundamentals Concepts, Drivers and Techniques*. Upper Saddle River, NJ, USA: Prentice-Hall, Jan. 2015.

[3] *EU General Data Protection Regulation*. Accessed: May 26, 2020. [Online]. Available: https://gdpr.eu/

[4] *Cambridge Dictionary*. Accessed: May 26, 2020. [Online]. Available: https://dictionary.cambridge.org

[5] K. Fujimura, H. Kuno, M. Terada, K. Matsuyama, Y. Mizuno, and J. Sekine, "Digital-ticket-controlled digital ticket circulation," in *Proc. 8th USENIX Secur. Symp.*, Washington, DC, USA, Aug. 1999, pp. 229–240.

[6] K. Verslype, B. De Decker, V. Naessens, G. Nigusse, J. Lapon, and P. Verhaeghe, "A privacy-preserving ticketing system," in *Proc. 22nd Annu. IFIP WG*, London, U.K., Jul. 2008, pp. 97–112.

[7] D. Quercia and S. Hailes, "MOTET: Mobile transactions using electronic tickets," in *Proc. 1st Int. Conf. Secur. Privacy for Emerg. Areas Commun. Netw. (SECURECOMM)*, Sep. 2005, pp. 374–383.

[8] H. Wang, Y. Zhang, J. Cao, and Y. Kambayahsi, "A global ticket-based access scheme for mobile users," *Inf. Syst. Frontiers*, vol. 6, no. 1, pp. 35–46, Mar. 2004.

[9] Y. Lei, A. Quintero, and S. Pierre, "Mobile services access and payment through reusable tickets," *Comput. Commun.*, vol. 32, no. 4, pp. 602–610, Mar. 2009, doi: 10.1016/j.comcom.2008.11.035.

[10] A. S. Amoli, M. Kharrazi, and R. Jalili, "2Ploc: Preserving privacy in location-based services," in *Proc. IEEE 2nd Int. Conf. Social Comput.*, Aug. 2010, pp. 707–712.

[11] G. Arfaoui, J.-F. Lalande, J. Traoré, N. Desmoulins, P. Berthomé, and S. Gharout, "A practical set-membership proof for privacy-preserving NFC mobile ticketing," *Proc. Privacy Enhancing Technol.*, vol. 2015, no. 2, pp. 25–45, Jun. 2015, doi: 10.1515/popets-2015-0019.

[12] A. Rupp, G. Hinterwälder, F. Baldimtsi, and C. Paar, "P4R: Privacy-preserving pre-payments with refunds for transportation systems," in *Proc. FC*, Okinawa, Japan, Apr. 2013, pp. 205–212, doi: 10.1007/978-3-642-39884-1_17.

[13] G. Arfaoui, G. Dabosville, S. Gambs, P. Lacharme, and J.-F. Lalande, "A privacy-preserving NFC mobile pass for transport systems," *ICST Trans. Mobile Commun. Appl.*, vol. 2, no. 5, p. e4, Dec. 2014, doi: 10.4108/mca.2.5.e4.

[14] S. J. Aboud, M. Al-Fayoumi, and M. Al-Fayoumi, "Efficient e-tickets fare scheme for time-typed services," *J. Internet Banking Commerce*, vol. 22, no. 1, pp. 1–20, 2017.

[15] A. Vives-Guasch, M.-M. Payeras-Capellà, M. Mut-Puigserver, J. Castella-Roca, and J.-L. Ferrer-Gomila, "A secure E-Ticketing scheme for mobile devices with near field communication (NFC) that includes exculpability and reusability," *IEICE Trans. Inf. Syst.*, vol. 95, no. 1, pp. 78–93, 2012, doi: 10.1587/transinf.E95.D.78.

[16] M. Milutinovic, K. Decroix, V. Naessens, and B. De Decker, "Privacy-preserving public transport ticketing system," in *Proc. 29th Annu. IFIP WG*, Fairfax, VA, USA, Jul. 2015, pp. 135–150.

[17] M. Mut-Puigserver, M. M. Payeras-Capellà, J.-L. Ferrer-Gomila, A. Vives-Guasch, and J. Castellà-Roca, "A survey of electronic ticketing applied to transport," *Comput. Secur.*, vol. 31, no. 8, pp. 925–939, Nov. 2012, doi: 10.1016/j.cose.2012.07.004.

[18] B. Patel and J. Crowcroft, "Ticket based service access for the mobile user," in *Proc. 3rd Annu. ACM/IEEE Int. Conf. Mobile Comput. Netw. MobiCom*, 1997, pp. 223–233.

[19] A.-R. Sadeghi, I. Visconti, and C. Wachsmann, "User privacy in transport systems based on RFID e-tickets," in *Proc. PiLBA*, Malaga, Spain, vol. 9, Oct. 2008, pp. 102–121.

[20] F. Kerschbaum, H. W. Lim, and I. Gudymenko, "Privacy-preserving billing for e-ticketing systems in public transportation," in *Proc. 12th ACM Workshop Privacy Electron. Soc. WPES*, 2013, pp. 143–154.

[21] J. Han, L. Chen, S. Schneider, H. Treharne, and S. Wesemeyer, "Privacy-preserving electronic ticket scheme with attribute-based credentials," *IEEE Trans. Dependable Secure Comput.*, early access, Sep. 12, 2019, doi: 10.1109/TDSC.2019.2940946.

[22] A. Vives-Guasch, M. Payeras-Capella, M. Mut-Puigserver, and J. Castellà-Roca, "E-ticketing scheme for mobile devices with exculpability," in *Proc. DPM*. Berlin, Germany: Springer, 2011, pp. 79–92, doi: 10.1007/978-3-642-19348-4_7.

[23] T. S. Heydt-Benjamin, H.-J. Chae, B. Defend, and K. Fu, "Privacy for public transportation," in *Proc. PET*. Berlin: Springer, 2006, pp. 1–19, doi: 10.1007/11957454_1.

[24] *Stræto App*. Accessed: May 26, 2020. [Online]. Available: https://www.straeto.is/is/um-straeto/straeto-appid

[25] M. Naor and M. Yung, "Universal one-way hash functions and their cryptographic applications," in *Proc. 21st Annu. ACM Symp. Theory Comput. STOC*, 1989, pp. 33–43.

[26] M. Bellare, R. Canetti, and H. Krawczyk, "Keying hash functions for message authentication," in *Proc. CRYPTO*, Santa Barbara, CA, USA, Aug. 1996, pp. 1–15.

[27] N. Asghar, "A survey on blind digital signatures," Dept. Combinatorics Optim., Univ. Waterloo, ON, Canada, Tech. Rep., 2011.

[28] D. Chaum, *Blind Signature System*. Boston, MA, USA: Springer, 1984, p. 153, doi: 10.1007/978-1-4684-4730-9_14.

[29] M. Abe and T. Okamoto, "Provably secure partially blind signatures," in *Proc. Advances in Cryptology—CRYPTO*, Santa Barbara, CA, USA, Aug. 2000, pp. 271–286.

[30] O. Goldreich, *Foundations of Cryptography: Volume 1, Basic Tools*. Cambridge, U.K.: Cambridge Univ. Press, 2001, pp. 223–228.

[31] F. Boudot, "Efficient proofs that a committed number lies in an interval," in *Advances in Cryptology—EUROCRYPT*. Berlin, Germany: Springer, May 2000, pp. 431–444, doi: 10.1007/3-540-45539-6_31.

[32] *TOR Project*. Accessed: May 26, 2020. [Online]. Available: https://www.torproject.org

RICARD BORGES was born in Mollerussa, Spain, in 1980. He received the M.Sc. degree in computer engineering from Rovira i Virgili University, Spain, in 2005. He is currently pursuing the Ph.D. degree in engineering and information technologies program with the Universitat de Lleida, Spain.

From 2005 to 2015, he worked with private companies as a Software Developer and Analyst. He is also responsible for the payment software of the European agricultural aid (FEADER), Generalitat de Catalunya, Lleida, Spain. His fields of interest include cryptography and information privacy.

FRANCESC SEBÉ was born in Lleida, Spain, in 1978. He received the M.Sc. degree in computer engineering from Rovira i Virgili University, Spain, in 2001, and the Ph.D. degree in telematics engineering from the Technical University of Catalonia, Spain, in 2003.

From 2003 to 2008, he was an Associate Professor with Rovira i Virgili University. He is currently a Full Professor with the Department of Mathematics, Universitat de Lleida, Spain. He is the author of more than 70 international publications. His fields of interest are cryptography and information privacy. He has participated in several European and Spanish funded research projects.