Noname manuscript No.
(will be inserted by the editor)

Design and Performance Evaluation of Two Approaches to Obtain Anonymity in Transferable Electronic Ticketing Schemes

M. Magdalena Payeras-Capellà · Macià Mut-Puigserver · Jordi Castellà-Roca · Julio Bondia-Barceló

the date of receipt and acceptance should be inserted later

Abstract Electronic tickets demonstrate, without the use of paper, the possession of the right to access or use of a service. In this scenario, the security and privacy achieved in the paper-based system must be preserved in the electronic ticketing systems. In addition to these requirements, the transferability of a ticket from one user to another (without involving a third party) is very useful but also generates other issues to be solved in terms of security and privacy. For example, the users that transfer the same ticket twice must be identified (it is a fraud), but the rest of users have to preserve their privacy (anonymity). In this article we present two proposals of an electronic ticketing system with anonymity and transferability, based on the use of RSA signatures or group signatures, respectively, thus presenting different levels of complexity and anonymity. Moreover, we have implemented both proposals and its performance has been evaluated. The results of this evaluation are useful in order to prove the viability of the proposals even when complex cryptography is used. The strength of the anonymity and the execution costs are the parameters to take into account to choose the more adequate proposal for a specific application.

Keywords e-ticketing, e-commerce, transferability, privacy, security, performance.

M. Magdalena Payeras-Capellà · Macià Mut-Puigserver
Dpt. Ciències Matemàtiques i Informàtica, Universitat de les Illes Balears, Carretera de Valldemossa, Km. 7,5, 07122, Palma, Spain
E-mail: mpayeras@uib.cat
E-mail: macia.mut@uib.cat

Jordi Castellà-Roca · Julio Bondia-Barceló
Dpt. Enginyeria Informàtica i Matemàtiques, UNESCO Chair in Data Privacy, Universitat Rovira i Virgili, Av. Països Catalans 26, E-43007, Tarragona, Spain
E-mail: jordi.castella@urv.cat
E-mail: julio.bondia@urv.cat

1 Introduction

The presence of information technologies is increasing progressively in our society. Users can access online services regardless of the place and time. For example, they can purchase a movie or theater ticket online. Nonetheless, in some cases, they have to print the ticket to have access to the service. In other words, the process is not completely electronic because a printed ticket is required. Thanks to the introduction of smartphones, all the processes can be performed electronically. These devices offer a good computational power, high storage capacity, and also different communication technologies, such as Near Field Communication (NFC). All these features can be available in a small device that allows mobility and flexibility, and which makes the system perfectly suitable for management, e-ticketing and e-payment schemes [5]. Electronic tickets can be defined as a representation of the owner's rights to act as a user of a service, thus preserving the same requirements than the ones offered in paper format. We would like to emphasize the following properties: anonymity and transferability.

In the same way as paper tickets, electronic tickets have different properties according to the services for which they are used. These services can be classified according to the anonymity offered. For instance, a plane e-ticket is non-anonymous: the identity of the passenger is a fixed parameter that is part of the e-ticket. In the case of e-tickets with revocable anonymity, the beneficiary can use the ticket as the owner of it but without any need for identification. This modality helps to avoid fraud related to the reuse of e-tickets. E-tickets with non-revocable anonymity are not linked to a user at all. The user who owns this e-ticket is the one who can use this service. In transferable tickets, several kinds of anonymity can be achieved, that is, weak or full depending on the features of the proposals.

Regarding the transferability property, there are several e-ticket sales and distribution companies that allow e-ticket transfer [1] [2]. Nonetheless, typical transfers of e-tickets are performed through a central service and are non-anonymous. We would like to transfer an electronic ticket in the same way that we can transfer a paper ticket, i.e. anonymously and without the participation of a central service.

We can find new systems by using cryptographic techniques that enable online e-ticket issue and verification, as in [4], [6], [9], [11], [12]. These actions are really important if users purchase the electronic tickets before their use, and not everywhere the e-ticket is able to be used with an online connection to a central database. In [13], a recent implementation of electronic tickets over mobile devices with NFC technology has been performed. Another example is the InMoDo system (Mobile Phone as a Ticket) [3], and it has been adopted by the Swedish national train company, among others.

In some concrete systems, the receiver of an electronic ticket uses smartcards to carry it. This is the case of Oyster card for public transport in London. This system was designed in order to make the scanners work independently when the central system connectivity was down.

However, users can not transfer their e-tickets to other users in [4], [12] and [13]. The transferability defined in [6] is carried out between transports systems. Users can use several transport systems (metro, bus, etc..) with the same e-ticket but they cannot transfer their e-tickets between them. Finally, the transferability provided in [9] and [11] does not allow the resolution of the conflict. Users can transfer the same e-ticket to several users (overspending) and they can prevent from being identified.

1.1 Contribution

A complete survey in this field can be found in [10]. The previous proposals, which have been already analyzed, use two different approaches to provide transferability to e-tickets: i) use a central service that synchronizes the transfer of e-tickets between users and does not allow revocable anonymity; ii) use a basic method of transfer without a complete conflict resolution system. In this case, it is not possible to identify the author of the overspending.

Thus, the main goal of our contribution is to preserve the security properties of (a) revocable anonymity, where the user's identity could be only revealed in case of misbehaviour, (b) transferability, where electronic tickets can be transferred as a resale or loan without the collaboration of a central service, and (c) short-term linkability, where the user can easily demonstrate that she is the same user when receiving the ticket and later when transferring it. This can be achieved by using different approaches. In [1], we presented a proposal that uses short-term linkability with group signatures. We now present two approaches and we compare both of them in relation to security and privacy (anonymity).

Both approaches, could allow to deploy transferable electronic ticketing in real scenarios, by fulfilling these security requirements. However, the anonymity degree provided by both approaches is different, since one offers strong anonymity, the other offers weak anonymity. Strong anonymity is achieved using more robust cryptography and a more complex protocol. For this reason we expect higher computational cost than in the proposal with weak anonymity. On the other hand, weak anonymity should be faster because it is based on lightweight cryptography.

We have implemented both proposals in order to evaluate its performance, trying to show if the additional computation can be executed in time for the viability of the system, thus proving whether the strong anonymity is possible in transferable e-ticketing.

1.2 Document organization

First, section 2 details a brief background explaining the cryptographic techniques used in the protocols. In section 3, we explain the two proposals with its desired properties, the entities and phases. The security analysis is performed in section 4, section 5 describes the implementations and the performance evaluation and finally, in section 6, we compare the approaches, state the conclusions and future work.

2 Background

We use the short group signature (BBS) scheme [3] in order to verify that a user is a correct member of a certain group of users. Next, we introduce the main definitions related to the BBS signature, either the group signatures scheme and the Zero-Knowledge Proof (ZKP) of the group signatures.

2.1 Group signatures scheme

In this section, we specify the procedures $(KeyGen_G, Sign_G, Verify_G, Open_G, SignLinkable_G, VerifyLinkable_G)$ to be further used in the protocol with their parameters. $KeyGen_G, Sign_G, Verify_G$ and $Open_G$ are constructed from the same BBS scheme [3]. Both $SignLinkable_G$ and $VerifyLinkable_G$ have also been constructed in [7]. Consider bilinear groups G_1 and G_2 with respective generators g_1 and g_2. Suppose that the SDH assumption [8] holds on (G_1, G_2), and that the

[1] http://www.ticketmaster.com/transfer
[2] https://www.e-ticket.lu
[3] http://inmodo.com/

Linear assumption [3] holds on G_1. The scheme uses a bilinear map $e : G_1 \times G_2 \rightarrow G_T$ and a hash function $H : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$. The public values are $g_1, u, v, h \in G_1$ and $g_2, w \in G_2$. Here $w = g_2^{\gamma}$ for some secret $\gamma \in \mathbb{Z}_p$. The functions are:

- $KeyGen_G(n)$. This algorithm takes a parameter n as input, which is the number of members of the group. The algorithm has the following steps:
  1. select $h \xleftarrow{R} G_1 \backslash \{1_{G_1}\}$ and $gmsk = (\xi_1, \xi_2)$, where $\xi_1, \xi_2 \xleftarrow{R} \mathbb{Z}_p^*$, and set $u, v \in G_1$ such that $u^{\xi_1} = v^{\xi_2} = h$;
  2. select $\gamma \xleftarrow{R} \mathbb{Z}_p^*$ and set $w = g_2^{\gamma}$; and
  3. generate for each user $\mathcal{U}_i$, $1 \leq i \leq n$, an SDH tuple (A_i, x_i) by performing: select $x_i \xleftarrow{R} \mathbb{Z}_p^*$ and set $A_i \leftarrow g_1^{1/(\gamma + x_i)}$. The parameter γ is the private master key of the group key issuer.
- $Sign_G(gpk, gsk[i], M)$. Given a group public key $gpk = (g_1, g_2, h, u, v, w)$, a private user's key $gsk[i] = (A_i, x_i)$ and a message $M \in \{0,1\}^*$, compute and output a signature of knowledge $\sigma = (T_1, T_2, T_3, c, s_\alpha, s_\beta, s_x, s_{\delta_1}, s_{\delta_2})$. Note that the tuple (T_1, T_2, T_3) is the linear encryption of A, that is: $(T_1, T_2, T_3) = (u^\alpha, v^\beta, Ah^{\alpha+\beta})$ for $\alpha, \beta \xleftarrow{R} \mathbb{Z}_p$. There are also some helper values $\delta_1 \leftarrow x\alpha$ and $\delta_2 \leftarrow x\beta$. The parameter c is the self-generated challenge (hash of the information in the commit information of the proof of knowledge). Finally, $(s_\alpha, s_\beta, s_x, s_{\delta_1}, s_{\delta_2})$ are the response values of the proof of knowledge.
- $Verify_G(gpk, M, \sigma)$. Given a group public key $gpk = (g_1, g_2, h, u, v, w)$, a message M and a group signature $\sigma = (T_1, T_2, T_3, c, s_\alpha, s_\beta, s_x, s_{\delta_1}, s_{\delta_2})$, verify that σ is a valid signature of the message.
- $Open_G(gpk, gmsk, M, \sigma)$. This algorithm is used in order to trace a signature to a concrete signer inside the group. It is only available for the group manager, as she is the holder of the $gmsk$ master key and knows all the pairs (A_i, x_i). Given a group public key $gpk = (g_1, g_2, h, u, v, w)$, the group master private key $gmsk = (\xi_1, \xi_2)$, a message M and a signature $\sigma = (T_1, T_2, T_3, c, s_\alpha, s_\beta, s_x, s_{\delta_1}, s_{\delta_2})$, it proceeds as follows. First, recover the user's A by performing $A \leftarrow T_3/(T_1^{\xi_1} \cdot T_2^{\xi_2})$. If the elements $\{A_i\}$ of the user's private keys are given to the group manager, then she can look up the user index corresponding to the identity A recovered from the signature.
- $SignLinkable_G(gpk, gsk[i], M', \sigma, \alpha, \beta)$. Given a group public key gpk, a private user's key $gsk[i]$, a message M', a previous signature σ, and the values α, β used for that signature, compute and output a signature σ'. In order to use this procedure correctly, it is defined as follows:
  - First use: standard $Sign_G(gpk, gsk[i], M)$ obtaining a group signature σ and using (α, β).
  - Further uses: $SignLinkable_G(gpk, gsk[i], M', \sigma, \alpha, \beta)$:
    1. use the same pair (α, β) producing the same linear encryption of A as in the first time: $(T_1, T_2, T_3) = (u^\alpha, v^\beta, Ah^{\alpha+\beta})$; and
    2. given a message M', sign the message and output a signature $\sigma' \leftarrow (T_1, T_2, T_3, c', s'_\alpha, s'_\beta, s'_x, s'_{\delta_1}, s'_{\delta_2})$, where $c' \leftarrow H(M', T_1, T_2, T_3, R'_1, R'_2, R'_3, R'_4, R'_5) \in \mathbb{Z}_p$.

  Note that the triviality of the several signatures are produced by the same user, as the information (T_1, T_2, T_3) is public in the same signature. In addition, the random values $(r_\alpha, r_\beta, r_x, r_{\delta_1}, r_{\delta_2})$ must be different from the previous times, that is: $({r_\alpha}' \neq r_\alpha, {r_\beta}' \neq r_\beta, {r_x}' \neq r_x, {r_{\delta_1}}' \neq r_{\delta_1}, {r_{\delta_2}}' \neq r_{\delta_2})$ in order not to reveal information.
- $VerifyLinkable_G(\sigma, \sigma')$. This algorithm takes two signatures σ and σ' as input and outputs $true$ or $false$ depending on whether the signatures have been produced by the same signer's pseudonym: $(T_1 \stackrel{?}{=} {T_1}', T_2 \stackrel{?}{=} {T_2}', T_3 \stackrel{?}{=} {T_3}')$.

2.2 ZKP of the group signatures scheme

In our proposal, both standard and linkable group signatures are used, as they enable to verify the internal message information as well as to verify that determined signatures related to the same event or e-ticket belong to the same *anonymous* user. Despite these advantages, the signatures are generated by the same user, and the verifications can be performed *offline*, that is, the verifier does not take a role during the signature generation, and the verification process therefore becomes critical. Then we detail the procedures $ZKP_GCommit$, $ZKP_GResponse$ and $ZKP_GVerify$:

- $ZKP_GCommit(M^*)$. This procedure is performed by the user that wants to demonstrate (prover) to another user (verifier) that she is the right holder of the ticket. Given a public group key $gpk = (g_1, g_2, h, u, v, w)$, a group private key for the user $gsk[i] = (A_i, x_i)$ and a signed message $M^* = (M, \sigma)$, where $\sigma = (T_1, T_2, T_3, c, s_\alpha, s_\beta, s_x, s_{\delta_1}, s_{\delta_2})$, it generates the commitment $m' = (T_1, T_2, T_3, R'_1, R'_2, R'_3, R'_4, R'_5)$ as output.
  1. we have to demonstrate the ownership of the values $(\alpha, \beta, x, \delta_1, \delta_2)$ that have been generated by the signature of M^*, keeping then the resulting values with the linear encryption of A: $(T_1, T_2, T_3) = (u^\alpha, v^\beta, Ah^{\alpha+\beta})$;
  2. the values ${r_\alpha}', {r_\beta}', {r_x}', {r_{\delta_1}}', {r_{\delta_2}}' \xleftarrow{R} \mathbb{Z}_p$ are selected and then the following values are generated:

   a) $R'_1 \leftarrow u^{r_\alpha{}'}$;
   b) $R'_2 \leftarrow v^{r_\beta{}'}$;
   c) $R'_3 \leftarrow e(T_3, g_2)^{r_x{}'} \cdot e(h, w)^{-r_\alpha{}' - r_\beta{}'} \cdot e(h, g_2)^{-r_{\delta_1}{}' - r_{\delta_2}{}'}$;
   d) $R'_4 \leftarrow T_1^{r_x{}'} \cdot u^{-r_{\delta_1}{}'}$; and
   e) $R'_5 \leftarrow T_2^{r_x{}'} \cdot v^{-r_{\delta_2}{}'}$.
3. the output $m' = (T_1, T_2, T_3, R'_1, R'_2, R'_3, R'_4, R'_5)$ is generated.

– $ZKP_G Response(m', c')$. Given a commitment m' where $m' = (T_1, T_2, T_3, R'_1, R'_2, R'_3, R'_4, R'_5)$ and a challenge c' given by the verifier, the prover generates the response $s' = (s_\alpha{}', s_\beta{}', s_x{}', s_{\delta_1}{}', s_{\delta_2}{}')$ where their values are given by:
1. $s_\alpha{}' \leftarrow r_\alpha{}' + c'\alpha$;
2. $s_\beta{}' \leftarrow r_\beta{}' + c'\beta$;
3. $s_x{}' \leftarrow r_x{}' + c'x$;
4. $s_{\delta_1}{}' \leftarrow r_{\delta_1}{}' + c'\delta_1$; and
5. $s_{\delta_2}{}' \leftarrow r_{\delta_2}{}' + c'\delta_2$.

– $ZKP_G Verify(m', c', s')$. Given a commitment m' where $m' = (T_1, T_2, T_3, R'_1, R'_2, R'_3, R'_4, R'_5)$, a challenge c' given by the verifier, and the response $s' = (s_\alpha{}', s_\beta{}', s_x{}', s_{\delta_1}{}', s_{\delta_2}{}')$ provided by the prover, the verifier checks
1. $u^{s_\alpha{}'} \stackrel{?}{=} T_1^{c'} \cdot R'_1$;
2. $v^{s_\beta{}'} \stackrel{?}{=} T_2^{c'} \cdot R'_2$;
3. $e(T_3, g_2)^{s_x{}'} \cdot e(h, w)^{-s_\alpha{}' - s_\beta{}'} \cdot e(h, g_2)^{-s_{\delta_1}{}' - s_{\delta_2}{}'} \stackrel{?}{=} (e(g_1, g_2)/e(T_3, w))^{c'} \cdot R'_3$;
4. $T_1^{s_x{}'} \cdot u^{-s_{\delta_1}{}'} \stackrel{?}{=} R'_4$; and
5. $T_2^{s_x{}'} \cdot v^{-s_{\delta_2}{}'} \stackrel{?}{=} R'_5$.

3 Description of the System

In this section, we describe our system proposal. First of all, the security requirements to achieve are briefly introduced in Section 3.1. Then, the details of the first approach are presented in Section 3.2 while the details of the second approach are presented in Section 3.3.

3.1 Requirements

We can classify e-ticket requirements into two categories [10]. On the one hand, we have security requirements, and on the other hand we have functional requirements for e-tickets. The main security requirements desired for our proposal are:

– Authenticity: the generated ticket is demonstrably genuine.
– Non-repudiation: the issuer of the ticket cannot deny its generation.
– Integrity: the ticket cannot be modified after its issue.
– Revocable anonymity: the ticket is anonymous, but if the user performs an unauthorized use, it can be then identified.
– Reusability: depending on the needs and features of the ticket it could be used multiple times or only once.
– Non-overspending: the ticket cannot be used more times than those established in the issue.
– Exculpability: the service provider cannot accuse the user of the ticket before its use.
– Transferability: the ticket can be transferred to other users, without losing any of the requirements previously stated.

The functional requirements considered are the following:

– Validity time: the ticket can be valid until an established time.
– Online/Offline: ticket verification can require a connection to the Internet or a centralized system for checking (Online), or this connection is not necessary (Offline).

3.2 Details of the protocol in the First Approach

This section describes the approach based on RSA signatures. The entities can be classified in two categories: basic actors and management. The basic actors are the User ($\mathcal{U}$), Issuer ($\mathcal{I}$) and Service provider ($\mathcal{P}$). The management is made up Registration Authority (RA) to assign a pseudonym to users, Certification Authority (CA) to create RSA public key certificates, and a Law Enforcement Authority (Ł) to intervene in case of conflict. The protocol presents seven phases: *Registration*, *Ticket Issue*, between $\mathcal{I}$ and $\mathcal{U}$; *Ticket Transfer (first time)*, between two users $\mathcal{U}_1$ and $\mathcal{U}_2$; *Ticket Transfer (following 'j' times)*, also between two users $\mathcal{U}_1$ and $\mathcal{U}_2$; *Ticket Verification (standard)*, between $\mathcal{U}$ and $\mathcal{P}$; *Ticket Verification (transferred)*, also between $\mathcal{U}$ and $\mathcal{P}$; and finally the *Revocation of anonymity* phase, which is only used in case of conflict, and where Ł and RA cooperate to revoke the user anonymity.

3.2.1 Registration

First, the user has to install the application of e-tickets on her smartphone. She must then register to make use of the various features that have been implemented in the e-ticket application. To do so, she contacts with the registration authority (RA) to receive an alias ($\mathsf{Pseu}_\mathcal{U}$). Afterwards, $\mathcal{U}$ gets a public key certificate from the certification authority (CA). Once this process has been completed, she can start using e-tickets to get access to the services offered. The registration protocol is carried out as follows:

1. $\mathcal{U}$:
   a) generates her RSA key-pair (public, private);

   b) creates a message with the user's information: $Credentials = (ID, CreditCardInformation, publicRSAkeyInfo, AdditionalInformation)$. The information in this message may vary depending on the final requirements of the implemented system;
   c) sends $Credentials$ to RA;
2. RA:
   a) verifies the user's identity and the user's information, and generates a new pseudonym ($\mathsf{Pseu}_{\mathcal{U}}$) for this $\mathcal{U}$.
   Should verification fail, RA stops the protocol;
   b) signs $\widehat{\mathsf{Pseu}_{\mathcal{U}}} = Sign_{RA}(\mathsf{Pseu}_{\mathcal{U}}||publicRSAkeyInfo)$; and
   c) sends $\mathsf{Pseu}_{\mathcal{U}}{}^{*} = (\mathsf{Pseu}_{\mathcal{U}}, \widehat{\mathsf{Pseu}_{\mathcal{U}}})$ to $\mathcal{U}$;
3. $\mathcal{U}$:
   a) sends $\mathsf{Pseu}_{\mathcal{U}}{}^{*}$ to CA in order to obtain a public key certificate;
4. CA:
   a) generates the user's certificate of her public RSA key: $cert_{\mathcal{U}}$; and
   b) sends $cert_{\mathcal{U}}$ to $\mathcal{U}$.
   The $cert_{\mathcal{U}}$ has the regular structure of an *X.509 Public Key Certificate*.

$\mathcal{U}$ has now registered and she has a valid $cert_{\mathcal{U}}$ to use the e-ticket system.

3.2.2 Ticket issue

Once we have purchased it, the e-ticket exists only as a digital record. The buyer ($\mathcal{U}$) receives the equivalent to a real ticket in her mobile device, which will be used to ensure access to the service. Technically, the electronic ticket is an information structure with identifying data (unique to each ticket) and descriptive data (indicating the contracted service). To get an e-ticket, $\mathcal{U}$ has to perform the following operations:

1. $\mathcal{U}$: sends a request to know what services are available: Req.
   The Req message is a signed message with the user's certificate ($cert_{\mathcal{U}}$) attached to it;
2. $\mathcal{I}$: verifies the user's credentials and sends the list of services to $\mathcal{U}$: List.
   Should verification fail, $\mathcal{I}$ stops the protocol;
3. $\mathcal{U}$:
   a) selects the service Sv from the List;
   b) generates the ticket request $\mathsf{V}^{*} = (\mathsf{V}, \widehat{\mathsf{V}})$, where $\mathsf{V} = (\mathsf{Sv}||n_{\alpha})$ and the signature request is $\widehat{\mathsf{V}} = Sign_{\mathcal{U}}(\mathsf{V})$;
   where $Sign()_{\mathcal{U}}$ is a regular RSA signature operation using the $\mathcal{U}$'s private key; and
   c) sends V^{*} to $\mathcal{I}$;
4. $\mathcal{I}$:
   a) generates the ticket information: $\mathsf{T} = (\mathsf{Sn}, \mathsf{V}^{*}, \mathsf{Ti}, \mathsf{Tv}, \mathsf{Tc})$, where the ticket includes Sn as the ticket serial number, V^{*} received from $\mathcal{U}$, Ti that is the date of issue, Tv is the validity time, and Tc is the terms and conditions. The terms and conditions include information such as: ticket issuer, price, and whether the ticket is transferable or not.
   b) signs the ticket: $\mathsf{T}^{*} = (\mathsf{T}, Sign_{\mathcal{I}}(\mathsf{T}))$;
   where this signature is the standard RSA signature with the $\mathcal{I}$'s private key; and
   c) sends the ticket T^{*} to $\mathcal{U}$;
5. $\mathcal{U}$:
   a) verifies the signature of T^{*}.

$\mathcal{U}$ has now received a valid ticket from $\mathcal{I}$ that gives the right to use the selected service.

3.2.3 Ticket Transfer (first time)

In this protocol, two users, $\mathcal{U}_1$ and $\mathcal{U}_2$, follow the steps described below to transfer an electronic ticket. The owner, $\mathcal{U}_1$, selects the desired ticket and generates a transfer request to a second user $\mathcal{U}_2$. After the execution of this protocol, $\mathcal{U}_2$ obtains the permission to use the ticket. The transfer works as follows:

1. $\mathcal{U}_1$: sends the ticket (T^{*}) to $\mathcal{U}_2$;
2. $\mathcal{U}_2$:
   a) verifies the information and signature of T^{*};
   b) generates a random value $n_{\lambda_0} \xleftarrow{R} \mathbb{Z}_p$;
   c) generates the first *transfer commitment* with the number of transferred times $k = 0$ and the agreed $price$ for the transfer: $c_{\beta_0} = H(n_{\lambda_0}||k = 0||price||\mathsf{flag_transfer})$;
   d) signs it: $\widehat{c}_{\beta_0} = (Sign_{\mathcal{U}}(c_{\beta_0}))$; and
   e) sends $c^{*}_{\beta_0} = (c_{\beta_0}, \widehat{c}_{\beta_0})$ to $\mathcal{U}_1$;
3. $\mathcal{U}_1$:
   a) verifies the information and signature of c_{β_0};
   b) generates a random value $n_{\beta_0} \xleftarrow{R} \mathbb{Z}_p$; and
   c) sends $C_{\beta_0} = Sign_{\mathcal{U}_1}(c^{*}_{\beta_0}||n_{\beta_0})$ to $\mathcal{U}_2$;
4. $\mathcal{U}_2$:
   a) verifies the tranfered information and signatures (C_{β_0} and $\mathsf{T}.\mathsf{V}^{*}$): The signatures must have been performed by the same user (in this case, by $\mathcal{U}_1$).

3.2.4 Ticket Transfer (following 'j' times)

1. $\mathcal{U}_1$: sends $(\mathsf{T}^{*}, C_{\beta_i}, c^{*}_{\beta_i}$ for $\{i = 0, ...(j-1)\})$ to $\mathcal{U}_2$;
2. $\mathcal{U}_2$:
   a) verifies the information inside the ticket and signature of T^{*};
   b) verifies that signatures $\mathsf{T}.\mathsf{V}^{*}$, and C_{β_i} for $i = \{0...(j-1)\}$ are chained properly;
   c) generates a random value $n_{\lambda_j} \xleftarrow{R} \mathbb{Z}_p$;

d) generates the new *transfer commitment* with the number of transferred times $k = j$ and the agreed $price$ for the transfer: $c_{\beta_j} = H(n_{\lambda_j}||k = j||price||$ flag_transfer);
e) signs it: $\widehat{c}_{\beta_j} = (Sign_{\mathcal{U}}(c_{\beta_j}))$; and
f) sends $c^*_{\beta_j} = (c_{\beta_j}, \widehat{c}_{\beta_j})$ to $\mathcal{U}_1$;

3. $\mathcal{U}_1$:
a) verifies the information and signature of $c^*_{\beta_j}$;
b) generates a random value $n_{\beta_j} \overset{R}{\leftarrow} \mathbb{Z}_p$; and
c) sends $C_{\beta_j} = Sign_{\mathcal{U}_1}(c^*_{\beta_j}||n_{\beta_j})$ to $\mathcal{U}_2$;
4. $\mathcal{U}_2$:
a) checks the correctness of the information
b) verifies the signature C_{β_j} and verifies that this signature and the signature $\widehat{c}_{\beta_j-1}$have been performed by the same user (in this case, by $\mathcal{U}_1$).

3.2.5 Ticket Verification (standard)

This protocol is used when no transfer has been performed since its issue. Here, $\mathcal{U}$ shows the ticket to $\mathcal{P}$ in order to be verified to receive the associated service. The protocol works as follows:

1. $\mathcal{U}$ sends the ticket T^* to $\mathcal{P}$;
2. $\mathcal{P}$:
a) verifies the information and signature of T^*; and
b) generates a random value $n_\gamma \overset{R}{\leftarrow} \mathbb{Z}_p$ and sends it back to $\mathcal{U}$;
3. $\mathcal{U}$
a) generates $\mathsf{Y}^* = (\mathsf{Y}, \widehat{\mathsf{Y}})$, where $\mathsf{Y} = (n_\gamma||\mathsf{T.Sn}||$ flag_spend_standard) and $\widehat{\mathsf{Y}} = Sign(\mathsf{Y})$; and
b) sends Y^* to $\mathcal{P}$;
4. $\mathcal{P}$:
a) verifies the signature of $\widehat{\mathsf{Y}}$ and checks that this signature and the signature $\mathsf{T}.\widehat{\mathsf{V}}$ are generated by the same user; and
b) stores $(\mathsf{T}^*, \mathsf{Y}^* = (\mathsf{Y}, \widehat{\mathsf{Y}}))$ in $\mathcal{P}$'s centralized database.

3.2.6 Ticket Verification (transferred)

The procedure to verify a transferred ticket is similar to the previous case (section 3.2.5). Additionally, $\mathcal{P}$ has to check the *transfer commitment chain*. In the following protocol, the ticket has been transferred $(j + 1)$ times:

1. $\mathcal{U}$ sends the ticket T^* to $\mathcal{P}$ together with all C_{β_i} for $i = \{0...j\}$ and ;
2. $\mathcal{P}$:
a) verifies the information and signature of T^*; and
b) verifies the signature chain C_{β_i} for $i = \{0...j\}$;
c) generates a random value $n_\gamma \overset{R}{\leftarrow} \mathbb{Z}_p$ and sends it back to $\mathcal{U}$;
3. $\mathcal{U}$
a) generates $\mathsf{Y}^* = (\mathsf{Y}, \widehat{\mathsf{Y}})$, where $\mathsf{Y} = (n_\gamma||\mathsf{T.Sn}||$ flag_spend_transferred) and $\widehat{\mathsf{Y}} = Sign(\mathsf{Y})$; and
b) sends Y^* to $\mathcal{P}$;
4. $\mathcal{P}$:
a) verifies that both signatures (Y^*, $c^*_{\beta_j}$) have been generated by the same user; and
b) stores $(\mathsf{T}^*, \mathsf{Y}^*, C_{\beta_i} \forall i = \{0...j\})$ in $\mathcal{P}$'s centralized database.

3.2.7 Revocation of Anonymity

The real identity of the user in this e-ticket system is hidden beneath a pseudonym ($\mathsf{Pseu}_\mathcal{U}$). Moreover, the RA could allow any user the right to have more than one $\mathsf{Pseu}_\mathcal{U}$ in order to make it more difficult to trace the behaviour of the users. In any case, the system pretends to achieve what is usually called *weak anonymity* (it is unfeasible for an attacker to identify the spender of an e-ticket in a transaction) [2] in contrast to *strong anonymity* (it is unfeasible for an attacker to decide whether two transactions are made by the same user or not).
In this section, the steps to revoke user anonymity are described:

1. $\mathcal{P}$ detects an overspending case: a single ticket has been spent more than once;
2. Case 1: the ticket has been spent more than once using the same $\mathsf{Pseu}_\mathcal{U}$
a) $\mathcal{P}$:
i. In this case, $\mathcal{P}$ has found two Y^* with the signature of a same $\mathcal{U}$ associated to a single ticket; then
ii. sends $(\mathsf{Y}^*_1, \mathsf{Y}^*_2)$ to a Law Enforcement Authority (Ł);
b) Ł:
i. verifies the received information;
ii. sends a request to RA to reveal the user identity of the $\mathcal{U}$ linked to the pseudonym of the signatures in $(\mathsf{Y}^*_1, \mathsf{Y}^*_2)$;
c) RA: reveals the user identity to Ł
3. Case 2: different users have spent the same e-ticket
a) $\mathcal{P}$:
i. In this case, $\mathcal{P}$ has two Y^* associated to a single T^*, but they are signed by different users. So, the $\mathcal{U}$ who has the right to spend it is the one who owns a longer signature chain: C_{β_i} for $i = \{0...j\}$; then
ii. The information associated to the ticket verification from both cases (two Y^* 's and T^*) is then sent to a Law Enforcement Authority (Ł);
b) Ł:
i. verifies the information received;

ii. sends a request to RA to reveal the user's identity of the $\mathcal{U}$ linked to the shorter signature chain;

c) RA: reveals the user identity to Ł

3.3 Details of the protocol in the Second Approach

This section describes the approach based on the short group signature (BBS) scheme [3]. There are four main entities in the system: User ($\mathcal{U}$), Issuer ($\mathcal{I}$), Service provider ($\mathcal{P}$) and Group Manager ($\mathcal{M}_\mathcal{G}$), which only interacts in case of conflict. The protocol has six phases.

3.3.1 Ticket issue

In this protocol, $\mathcal{U}$ receives a valid ticket from $\mathcal{I}$ in order to be further used. The protocol works as follows:

1. $\mathcal{I}$ generates and sends a random value $n_\alpha \overset{R}{\leftarrow} \mathbb{Z}_p$;
2. $\mathcal{U}$:
   a) selects the service Sv;
   b) generates $\mathsf{V}^* = (\mathsf{V}, \widehat{\mathsf{V}})$, where $\mathsf{V} = (\mathsf{Sv}||n_\alpha||\mathsf{flag_issue})$ and $\widehat{\mathsf{V}} = Sign_G(gpk, gsk[i], \mathsf{V})$ is the group signature;
   c) sends V^* to $\mathcal{I}$;
3. $\mathcal{I}$:
   a) verifies the group signature: $Verify_G(gpk, \mathsf{V}, \widehat{\mathsf{V}})$;
   b) generates the ticket information: $\mathsf{T} = (\mathsf{Sn}, \mathsf{V}, \mathsf{Ti}, \mathsf{Tv}, \mathsf{Tc})$, where the ticket includes V received from $\mathcal{U}$, Sn as the ticket serial number, Ti as the date of issue, Tv as the validity time, and Tc as the terms and conditions;
   c) signs the ticket: $\mathsf{T}^* = Sign_\mathcal{I}(\mathsf{T})$, where this signature could be a standard RSA-like signature; and
   d) sends the ticket T^* to $\mathcal{U}$;
4. $\mathcal{U}$ verifies the signature of T^*.

3.3.2 Ticket transfer (first time)

In this protocol, $\mathcal{U}_1$ transfers the original ticket to $\mathcal{U}_2$ by giving the permission to use it with a group signature which is linked to the commitment of the ticket V^* issued. The protocol works as follows:

1. $\mathcal{U}_1$:
   a) generates a commitment $m_{\beta_0} = ZKP_GCommit(\mathsf{T}^*)$; and
   b) sends the commitment and the ticket $(m_{\beta_0}||\mathsf{T}^*)$ to $\mathcal{U}_2$;
2. $\mathcal{U}_2$:
   a) verifies the information and signature of T^*: $Verify_\mathcal{I}(\mathsf{T}^*)$;
   b) verifies the group signature: $Verify_G(gpk, \mathsf{T.V}, \mathsf{T.}\widehat{\mathsf{V}})$;
   c) generates a random value $n_{\lambda_0} \overset{R}{\leftarrow} \mathbb{Z}_p$; and
   d) generates the first *challenge* with the number of transferred times $k = 0$ and the $price$ for the transfer agreed: $c_{\beta_0} = H(n_{\lambda_0}||k = 0||price||\mathsf{flag_transfer})$ and sends it to $\mathcal{U}_1$;
3. $\mathcal{U}_1$:
   a) generates the response $s_{\beta_0} = ZKP_GResponse(m_{\beta_0}, c_{\beta_0})$;
   b) generates a random value $n_{\beta_0} \overset{R}{\leftarrow} \mathbb{Z}_p$; and
   c) sends $(s_{\beta_0}||n_{\beta_0})$ to $\mathcal{U}_2$;
4. $\mathcal{U}_2$:
   a) verifies the response: $ZKP_GVerify(m_{\beta_0}, c_{\beta_0}, s_{\beta_0})$; and
   b) generates $\mathsf{W_0}^* = (\mathsf{W_0}, \widehat{\mathsf{W_0}})$, where $\mathsf{W_0} = (n_{\beta_0}||\mathsf{T}^*||\mathsf{flag_transfer})$ and its group signature $\widehat{\mathsf{W_0}} = Sign_G(gpk, gsk[i], \mathsf{W_0})$ and sends it to $\mathcal{U}_1$;
5. $\mathcal{U}_1$:
   a) verifies the group signature: $Verify_G(gpk, \mathsf{W_0}, \widehat{\mathsf{W_0}})$; and
   b) generates $\mathsf{X_0}^* = (\mathsf{X_0}, \widehat{\mathsf{X_0}})$, where $\mathsf{X_0} = \mathsf{W_0}^*$ and $\widehat{\mathsf{X_0}}$ is a group signature which is linkable only to V^*: $\widehat{\mathsf{X_0}} = SignLinkable_G(gpk, gsk[i], \mathsf{W_0}^*, \widehat{\mathsf{V}}, \alpha, \beta)$, and sends it to $\mathcal{U}_2$;
6. $\mathcal{U}_2$:
   a) verifies the group signature: $Verify_G(gpk, \mathsf{X_0}, \widehat{\mathsf{X_0}})$ and
   b) verifies that the two signatures have been performed by the same user: $VerifyLinkable_G(\mathsf{T.}\widehat{\mathsf{V}}, \widehat{\mathsf{X_0}})$. $\mathsf{X_0}^*$ works as a transfer agreement of the ticket of the user $\mathcal{U}_1$ to the user $\mathcal{U}_2$.

3.3.3 Ticket transfer (following 'k' times)

In this protocol, $\mathcal{U}_1$ transfers the (already transferred) ticket $\mathsf{X_{k-1}}$ to $\mathcal{U}_2$ by giving the permission to use it with a group signature which is linked to the commitment of the ticket previously received. The protocol works as follows:

1. $\mathcal{U}_1$:
   a) generates a commitment $m_{\beta_k} = ZKP_GCommit(\mathsf{X_{k-1}})$; and
   b) sends the commitment and ticket $(m_{\beta_k}||\mathsf{X_{k-1}})$ to $\mathcal{U}_2$;
2. $\mathcal{U}_2$:
   a) verifies the information and signature of T^*: $Verify_\mathcal{I}(\mathsf{T}^*)$, the group signature: $Verify_G(gpk, \mathsf{V}, \widehat{\mathsf{V}})$, and the linkability of the two group signatures of the beginning of the first transfer: $VerifyLinkable_G(\widehat{\mathsf{X_0}}, \mathsf{T.}\widehat{\mathsf{V}})$;
   b) for each transfer $\forall i \in [0, k)$, verifies the group signatures $Verify_G(gpk, \mathsf{X_i}, \widehat{\mathsf{X_i}})$ and $Verify_G(gpk, \mathsf{W_i}, \widehat{\mathsf{W_i}})$,

and checks the linkability of $VerifyLinkable_G(\widehat{\mathsf{X_i}}, \widehat{\mathsf{W_{i-1}}})$;

c) generates a random value $n_{\lambda_k} \overset{R}{\leftarrow} \mathbb{Z}_p$; and

d) generates the first *challenge*: $c_{\beta_k} = H(n_{\lambda_k}||k||price||\mathsf{flag_transfer})$ (with the price of the transfer agreed) and sends it to $\mathcal{U}_1$;

3. $\mathcal{U}_1$:

a) generates the response $s_{\beta_k} = ZKP_GResponse(m_{\beta_k}, c_{\beta_k})$;

b) generates a random value $n_{\beta_k} \overset{R}{\leftarrow} \mathbb{Z}_p$; and

c) sends $(s_{\beta_0}||n_{\beta_0})$ to $\mathcal{U}_2$;

4. $\mathcal{U}_2$:

a) verifies the response: $ZKP_GVerify(m_{\beta_k}, c_{\beta_k}, s_{\beta_k})$; and

b) generates $\mathsf{W_k}^* = (\mathsf{W_k}, \widehat{\mathsf{W_k}})$, where $\mathsf{W_k} = (n_{\beta_k}||\mathsf{X_{k-1}}||\mathsf{flag_transfer})$ and its group signature is $\widehat{\mathsf{W_k}} = Sign_G(gpk, gsk[i], \mathsf{W_k})$, and sends it to $\mathcal{U}_1$;

5. $\mathcal{U}_1$:

a) verifies the group signature: $Verify_G(gpk, \mathsf{W_k}, \widehat{\mathsf{W_k}})$; and

b) generates $\mathsf{X_k}^* = (\mathsf{X_k}, \widehat{\mathsf{X_k}})$, where $\mathsf{X_k} = \mathsf{W_k}^*$ and $\widehat{\mathsf{X_k}}$ is a group signature which is linkable only to $\mathsf{W_{k-1}}^*$: $\widehat{\mathsf{X_k}} = SignLinkable_G(gpk, gsk[i], \mathsf{W_k}^*, \widehat{\mathsf{W_{k-1}}}, \alpha, \beta)$, and sends it to $\mathcal{U}_2$;

6. $\mathcal{U}_2$:

a) verifies the group signature: $Verify_G(gpk, \mathsf{X_k}, \widehat{\mathsf{X_k}})$; and

b) verifies that the two signatures have been performed by the same user: $VerifyLinkable_G(\widehat{\mathsf{X_k}}, \widehat{\mathsf{W_{k-1}}})$. $\mathsf{X_k}^*$ works as a transfer agreement of the ticket of the user $\mathcal{U}_1$ to the user $\mathcal{U}_2$.

3.3.4 Ticket verification (standard)

This protocol is used when no transfer has been performed since its issue. Here, $\mathcal{U}$ shows the ticket to $\mathcal{P}$ in order to be verified to receive the associated service. The protocol works as follows:

1. $\mathcal{U}$ sends the ticket T^* to $\mathcal{P}$;
2. $\mathcal{P}$:

a) verifies the information and signature of T^*; and

b) generates a random value $n_\gamma \overset{R}{\leftarrow} \mathbb{Z}_p$ and sends it back to $\mathcal{U}$;

3. $\mathcal{U}$ generates $\mathsf{Y}^* = (\mathsf{Y}, \widehat{\mathsf{Y}})$, where $\mathsf{Y} = (n_\gamma||\mathsf{T.Sn}||\mathsf{flag_spend_standard})$ with a group signature which is linkable only to V^*: $\widehat{\mathsf{Y}} = SignLinkable_G(gpk, gsk[i], \mathsf{Y}, \widehat{\mathsf{V}}, \alpha, \beta)$;
4. $\mathcal{P}$:

a) verifies the group signature: $Verify_G(gpk, \mathsf{Y}, \widehat{\mathsf{Y}})$ and verifies that the two signatures are generated by the same user: $VerifyLinkable_G(\mathsf{T.}\widehat{\mathsf{V}}, \widehat{\mathsf{Y}})$; and

b) stores $\mathsf{T.Sn}$ in $\mathcal{P}$'s centralized database.

3.3.5 Ticket verification (transferred)

This protocol is used when some transfer has been performed since its issue. Here, $\mathcal{U}$ shows the ticket to $\mathcal{P}$ in order to be verified and to receive the associated service. The protocol works as follows:

1. $\mathcal{U}$ sends the transferred ticket $\mathsf{X_k}^*$ to $\mathcal{P}$;
2. $\mathcal{P}$:

a) verifies the information and signature of $\mathsf{X_k}^*$: $Verify_{\mathcal{I}}(\mathsf{T}^*)$ and $Verify_G(gpk, \mathsf{X_k}, \widehat{\mathsf{X_k}})$. The service provider $\mathcal{P}$ can detect whether the ticket has been transferred or not depending on its content;

b) verifies that the two signatures which are included in the ticket have been generated by the same user: $VerifyLinkable_G(\widehat{\mathsf{X_0}}, \mathsf{T.}\widehat{\mathsf{V}})$.

c) for all the transfers, $\forall i \in [0, k]$: verifies all the group signatures $Verify_G(gpk, \mathsf{X_i}, \widehat{\mathsf{X_i}})$ and $Verify_G(gpk, \mathsf{W_i}, \widehat{\mathsf{W_i}})$, and also the linkability of $\mathsf{W_{i-1}}^*$: $VerifyLinkable_G(\widehat{\mathsf{X_i}}, \widehat{\mathsf{W_{i-1}}})$ where needed; and

d) generates a random $n_\gamma \overset{R}{\leftarrow} \mathbb{Z}_p$ and sends it back to $\mathcal{U}$;

3. $\mathcal{U}$ generates $\mathsf{Y}^* = (\mathsf{Y}, \widehat{\mathsf{Y}})$, where $\mathsf{Y} = (n_\gamma||\mathsf{T.Sn}||\mathsf{flag_spend_transferred})$ and its linkable group signature to $\mathsf{W_k}^*$ as follows: $\widehat{\mathsf{Y}} = SignLinkable_G(gpk, gsk[i], \mathsf{Y}, \widehat{\mathsf{W_k}}, \alpha, \beta)$;
4. $\mathcal{P}$:

a) verifies the group signature: $Verify_G(gpk, \mathsf{Y}, \widehat{\mathsf{Y}})$, and verifies that the two signatures are generated by the same user: $VerifyLinkable_G(\widehat{\mathsf{W_k}}, \widehat{\mathsf{Y}})$, where the ticket receiver then has to demonstrate that she is the same user both in the transfer and verification phases; and

b) stores $\mathsf{T.Sn}$ in the $\mathcal{P}$'s centralized database.

3.3.6 Revocation of anonymity

This protocol follows the same procedure described in section 3.2.7. To spend an e-ticket, any user has to do a new signature at the 3rd step of the ticket verification phase (in both cases: sections 3.3.4 and 3.3.5). In case of controversy (such as an e-ticket overspending case), the group manager $\mathcal{M}_\mathcal{G}$ could take part in the resolution of the controversy and could revoke the anonymity of the signer that misbehaved by calling the $Open_G$ procedure.

4 Security and transferability of the System

The security analysis is organized in four propositions that state the security features of the two approaches. The respective claims then discuss and provide evidence to support the propositions. The security arguments are presented for each approach if necessary. This discussion does not provide any demonstration of the security of the cryptographic primitives and does not pretend to be a formal analysis of the security of the protocol, but it substantiates the security properties of the protocol. Moreover, the results of this discussion will provide information to compare the benefits of each approach.

Proposition 1 *Both proposals of anonymous transferable e-ticketing preserve authenticity, non-repudiation and integrity of the e-ticket.*

Claim 1 *It is computationally unfeasible to make a new fraudulent e-ticket.*
Security Argument. First approach. A valid e-ticket has the form $\mathsf{T}^* = Sign_{\mathcal{I}}(\mathsf{T})$. *The* $\mathcal{I}$ *generates the ticket information:* $\mathsf{T} = (\mathsf{Sn}, \mathsf{V}^*, \mathsf{Ti}, \mathsf{Tv}, \mathsf{Tc})$ *and signs the ticket:* $\mathsf{T}^* = Sign_{\mathcal{I}}(\mathsf{T})$, *where this signature is the standard RSA signature with the* $\mathcal{I}$*'s private key.*
Security Argument. Second approach. A valid e-ticket has the form $\mathsf{T}^* = (\mathsf{T}, \mathsf{Sign}_{\mathcal{I}}(\mathsf{T}))$.
In both cases, the first step that $\mathcal{P}$ *does when an e-ticket is received is the verification of the signature. The Ticket Verification protocol will continue only if this verification is positive; otherwise,* $\mathcal{P}$ *refuses* $\mathcal{U}$*'s request. Thus, making a new fraudulent valid e-ticket would be equivalent to breaking the signature scheme, which would be computationally unfeasible as we have supposed that* $\mathcal{I}$ *uses a secure signature scheme.*

Claim 2 *The issuer* $\mathcal{I}$ *cannot deny the emission of a valid e-ticket.*
Security Argument. A valid e-ticket has $\mathcal{I}$*'s signature, and the signature scheme used is secure. Consequently, the identity of the issuer is associated to the ticket i.e. the signature is a non-repudiation piece of evidence of her origin.*

Claim 3 *The content of the e-ticket cannot be modified. Security Argument. If someone modifies the content of the ticket, a new* $\mathcal{I}$*'s signature then has to be generated over the modified content; otherwise, the e-ticket will not pass the verification i.e. it will not be valid. Again, if it is computationally unfeasible to forge* $\mathcal{I}$*'s signature, it is unfeasible to modify the content of the e-ticket.*

Proposition 2 *The e-ticketing systems described in §3.2 and §3.3 are anonymous. The offered service is revocable anonymous.*

Claim 4 *The protocol to get an e-ticket is anonymous.*
Security Argument. First Approach. After obtaining her credentials, $\mathcal{U}$ *contacts the* $\mathcal{I}$ *in order to receive the e-ticket.* $\mathcal{U}$ *attaches the credentials to the request. The request is signed using a regular RSA signature using* $\mathcal{U}$*'s private key. Since the credentials are issued by the credentials issuer,* $\mathcal{I}$ *does not know the identity of* $\mathcal{U}$*. However, if* $\mathcal{U}$ *uses its credentials in several requests,* $\mathcal{I}$ *can link these requests. So the tickets are anonymous but linkable until* $\mathcal{U}$ *renews its credentials. For this reason, the anonymity achieved by this approach is a weak anonymity.*
Security Argument. Second Approach. The user establishes a connection with the ticket issuer $\mathcal{I}$ *in order to receive the e-ticket. Additionally, the user does not use any personal authentication method to get the e-ticket.* $\mathcal{I}$ *generates and sends the e-ticket to* $\mathcal{U}$ *if she accredits to be a member of the group of users by producing a valid group signature over a challenge sent by* $\mathcal{I}$*. The* $Verify_G(\mathsf{V})$ *procedure performed by* $\mathcal{I}$ *cannot identify the user due to the group signatures properties. For this reason, the anonymity achieved by this approach is strong anonymity.*

Claim 5 *An e-ticket has revocable anonymity.*
Security Argument. First Approach. The e-ticket is generated by $\mathcal{I}$*, who only knows the user's credentials as we have discussed in the previous claim. The only item inside the e-ticket that can be used to identify the user is the ticket request* $\mathsf{V}^* = (\mathsf{V}, \widehat{\mathsf{V}})$, *where* $\mathsf{V} = (\mathsf{Sv}||n_\alpha)$ *and the signature request is* $\widehat{\mathsf{V}} = Sign_{\mathcal{U}}(\mathsf{V})$. *Since the proposal presents weak anonymity, the* RA *can reveal the user's identity by using this information included in the e-ticket if it is required by Ł.*
Security Argument. Second Approach. A valid e-ticket does not have any information related to the user's identity. The e-ticket is generated by $\mathcal{I}$*, who does not know the user's identity as we have discussed in the previous claim. The only item inside the e-ticket that can identify the user is the group signature* V*, and only the group manager* $\mathcal{M}_{\mathcal{G}}$ *can reveal this information by performing* $Open_G(gpk, gmsk, M, \sigma)$*, because it is the only entity that knows* $gmsk$*. Therefore, e-tickets can be spent anonymously but anonymity can be revoked by* $\mathcal{M}_{\mathcal{G}}$*.* $\mathcal{M}_{\mathcal{G}}$ *plays the role of a trusted third party, thus it will only do that by law enforcement.*

Claim 6 *In spite of e-ticket anonymity, a fake user cannot spend an e-ticket impersonating another user. Security Argument. First Approach. In order to spend the e-ticket, the legitimate* $\mathcal{U}$ *has to prove the ownership of the e-ticket with the use of a pair of RSA signatures. The provider verifies the information and the signature of* $\mathsf{T}^* = Sign_{\mathcal{I}}(\mathsf{T})$. *In the standard verification, the provider generates a random value and the user signs a new element containing the random number using her private RSA key. Finally, the provider checks whether the signature included in the e-ticket and the new one are generated by the same user. If this is not the*

case, the e-ticket will not be accepted. In the verification of transferred tickets, a similar procedure is used. See proposition 4.
Security Argument. Second Approach. In order to spend the e-ticket, the legitimate $\mathcal{U}$ has to prove the ownership of the e-ticket with the use of a linkable signature Y with the element V, placed inside the e-ticket. An illegitimate $\mathcal{U}$ cannot perform the $SignLinkable_G$ operation properly: the fraud will be detected because the $VerifyLinkable_G()$ operation performed in the ticked verification protocol will warn $\mathcal{P}$ about this impersonation attack.

Proposition 3 *The protocol controls overspending.*

Claim 7 *If the ticket is only validated by one $\mathcal{P}$, the verification can be offline and $\mathcal{P}$ can control any overspending attempt.*
Security Argument. In this case, the security argument is the same for both approaches. $\mathcal{P}$ maintains a database with the serial numbers of the e-tickets already validated (i.e used). $\mathcal{P}$ can check both the issuer's signature and whether the e-ticket has not been spent before by using the information stored in the database. So the provider does not need to contact any party during the validation of an e-ticket.

Claim 8 *If the ticket is validated with several providers, all $\mathcal{P}$s must then be connected and must share a database of spent tickets.*
Security Argument. In this case, the security argument is similar for both approaches. The set of providers maintains a shared database with the serial numbers of the e-tickets that have been already validated. The contents of this database are used by the providers to decide whether they accept and validate a new ticket. So the provider does not need a connection with the issuer during the verification of an e-ticket, but the set of providers must share a database instead. If a persistent connection is not available, the overspending can then be detected and the identity of the overspender can be revealed by the group manager through the $Open_G(gpk, gmsk, M, \sigma)$ procedure in the second approach, and the credentials issuer in the first approach. Thus, the verification can be offline.

Proposition 4 *Users can transfer their e-ticket to other users by using the proposed scheme. The transferability operation among users preserves the security properties no matter how many transfers of the e-ticket have been made.*

Claim 9 *A transferred e-ticket can guarantee authenticity, non-repudiation and integrity properties as a non-transferred e-ticket.*
Security Argument. First Approach. For each transfer, a new transfer commitment is added to the ticket, including the number of transfers and the agreed price for the transfer, and it is signed by the user. Since a new RSA signature is added to the e-ticket, the properties of the signature keep authenticity, non-repudiation and integrity of the transferred e-ticket since the discussion of proposition 1 is already valid.
Security Argument. Second Approach. During a transfer, the format of the e-ticket is not substantially altered. Only a new group signature of the new owner is added to the e-ticket so that the properties of the signature keep authenticity, non-repudiation and integrity of the transferred e-ticket, since the discussion of proposition 1 is already valid.

Claim 10 *A transferred e-ticket preserves the anonymity of its owner.*
Security Argument. First Approach. During a transfer, the owner of the e-ticket signs the transfer commitment using her credentials. The receiver of the transferred e-ticket checks that the signature has been generated by the same user who obtained the ticket during the issue or a previous transfer. The identity of the signer is protected by the anonymous credential, so the weak anonymity is preserved during the transfer phase. The receiver of the ticket also signs a commitment in a similar way as in the issue of the ticket, so the anonymity of a transferred ticket is the same anonymity of an issued ticket, in this case weak anonymity, due to the possibility of linkage.
Security Argument. Second Approach. During a transfer, the owner of the e-ticket proves its ownership with a ZKP operation in order not to disclose her identity. The receiver of the transferred e-ticket includes a new group signature in it, so her identity is similarly protected in the same way, as we see in step 4 of the ticket verification protocol (transferred version).

Claim 11 *A transferred ticket cannot be overspent.*
Security Argument. First Approach. The overspending detection procedure described in proposition 3 is also valid for transferred e-tickets as well as the anonymity revocation which can be made through the credentials issuer because this property relies on the verification made by $\mathcal{P}$ on the serial number of the e-ticket stored in the database. The transfer of any e-ticket does not change its serial number and the provider receives the transfer commitment chain that includes all the signatures, and it can easily identify the user that has overspent the ticket.
Security Argument. Second Approach. The overspending detection procedure described in proposition 3 is also valid for transferred e-tickets as well as the anonymity revocation which can be made using $Open_G(gpk, gmsk, M, \sigma)$, because this property relies on the verification made by $\mathcal{P}$ on the serial number of the e-ticket stored in the database. The transfer of any e-ticket does not change its serial number.

5 Implementation and performance evaluation of both approaches

There are several important factors to consider when we design an e-ticketing system that should be usable in practice. The response time is one of them. Thus, we have implemented both protocols and we present some results regarding their time performance.

In Section 5.1, 5.2 and 5.3, we describe the developed components. Finally, we present the obtained results with the costs in the user side (Section 5.4).

5.1 Implementation

To implement the e-ticketing system in both versions (*RSA signature* and *Group Signature*), we have followed these requirements: general requirements (section 5.1.1), functional requirements (section 5.1.2), and technical requirements (section 5.1.3). The requirements of these sections are related to the way of developing the system implementation.

5.1.1 General requirements

The basic security requirements to be met by our implementations of the electronic ticketing system are *anonymity, portability* and *integrity*. The system must be fraud resistant, so it has to ensure that the user is going to be identified if she commits fraud. In this case, a Trusted Third Party will revoke the anonymity of the user. The operations available in the system must also be secure and ensure confidentiality and authenticity in communications between the different entities involved in the protocol.

5.1.2 Functional Requirements

Functional requirements are focused on the usability of the application and the management of electronic tickets by the user, since users must be satisfied with their experience.

The system based on group signatures has a higher computational cost, whereas the system implemented with RSA signatures has faster operations that do not add any remarkable delay, as we can see in the section 5.4.

Computation carried out in the application must be completely transparent to the user. Also, we have developed an intuitive GUI so that users know exactly what to do and how to manage their tickets in any operation.

5.1.3 Technical Requirements

We have used the JAVA programming language to implement the e-ticketing system. This language offers a wide variety of open source libraries. The basic information is represented in XML, with which we have an efficient access to the information stored within a message.

Both implementations have two distinct parts: the client application (implemented in Android, where users can get new tickets, manage, transfer and/or consume these tickets) and the cloud side, which implements the security protocol and the overall management of tickets.

The entities at the cloud side created to implement the e-ticketing system are:

- *CA and Validation Service*: On the one hand, the *CA* service is responsible for generating the credentials of each user and each ticket. It is also responsible for authenticating user credentials (certificate) and tickets (owner signature). On the other hand, the *Validation Service* has to check whether a fraud has been committed or not.
- *Register and Credentials Provider*: this service is responsible for registering new users. Also, it sends the new credentials (issued by the *CA*) to users.
- *Ticket Issuer*: this service allows users to obtain new tickets. Before issuing the tickets, it checks the user's authenticity.
- *Ticket Consumer*: this application gives a reliable service in exchange of an e-ticket. It verifies the user's certificate and signature before allowing the ticket spending. If a fraud is detected, the user's anonymity can be revoked.

These entities are merely external web services hosted in the cloud. In our case, we have used two different platforms: in case on the RSA signatures implementation we have used the *OpenShift* platform by *RedHat* (*OpenShift* is a platform as a service product from Red Hat. It supports applications such as web programming cartridges[4]) and in case of the Group Signatures implementation we have used the *App Engine* from Google [5]. We have developed these programmes in Java in all the cases, and databases are implemented in SQL.

Finally, Android has been chosen to make the mobile e-ticketing application. The most important reasons to choose this language are, on the one hand, the specific libraries that are available for both NFC and Bluetooth communications, both required to exchange tickets. On the other hand, the cryptographic libraries JPBC and BouncyCastle, which allowed us to programme the encryption operations and create, manage and verify certificates. Figure 1 depicts the system framework.

5.2 RSA Implementation

In this section, we would like to highlight some of the aspects of the implementation of the external entities in our e-ticketing system using the RSA signature scheme.

[4] https://www.openshift.com

[5] https://cloud.google.com/appengine/docs/whatisgoogleappengine

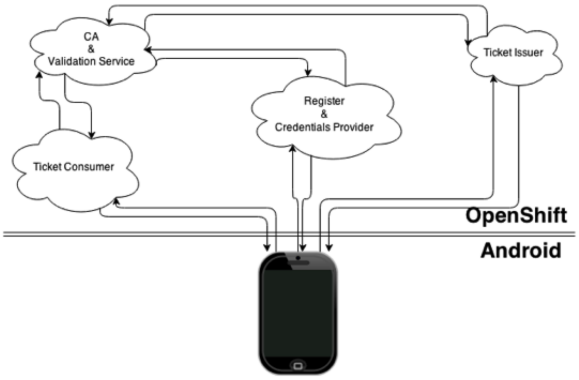


Fig. 1 E-ticketing System Framework

```
<Signature>
  <SignedInfo>
    <SignatureMethod />
    <CanonicalizationMethod />
    <Reference>
       <Transforms>
       <DigestMethod>
       <DigestValue>
    </Reference>
    <Reference /> etc.
  </SignedInfo>
  <SignatureValue />
  <KeyInfo />
  <Object />
</Signature>
```

Fig. 2 XML structure of a XMLDSig Digital Signature

5.2.1 Register and Credentials Provider

Digital signatures are used to guarantee the authentication of the messages exchanged. The structure of these messages follows the pattern of an XMLDSIG message (figure 2). The organization of such messages is depicted in figure 2. The tags of this message structure are:

- *SignedInfo*: It refers and contains the signed data.
- *SignatureMethod* and *CanonicalizationMethod*: They are used by *SignatureValue* and they are included to prevent tampering.
- One or more *Reference* elements are included to specify the resource that is being signed and any transformation that it is going to be applied to the resource before signing:
  - *DigestMethod* specifies the hash algorithm.
  - *DigestValue* contains the result of applying the hash function.
- *SignatureValue*: It contains the result of the signature operation over the *SignedInfo* element, encoded using BASE64 scheme. *CanonicalizationMethod* specifies the signature algorithm.
- *KeyInfo*: It allows the signer to provide slots with the key that validates the signature, usually in the form of one or more X.509 digital certificates.
- *Object*: contains the signed data in case of an enveloped signature.

5.2.2 Ticket Issuer

The Ticket Issuer is a web service that uses a database with one table of *Ticket* objects. The objects have the serial number of each electronic ticket and the rest of its content. The information is encoded in BASE64. The User ID (the owner of the ticket) is stored together with each e-ticket.

Ticket signatures are also created with the XMLDSIG scheme. For example, the issuer signs with its private key (RSA-2048 key) and includes the XML digital certificate with its public key to verify that the ticket has been truly signed by this entity. Once the digital signature has been generated, it is added to the ticket.

5.2.3 CA and Validation Service

The database used by this web service has a table to store the spent tickets. The database stores the serial number of each spent ticket, its content (encoded in BASE64) and the User ID who has spent the ticket. This table is essential to detect any possible fraud of double-spending. Note that the electronic ticket, depending on the number of times that has been transferred (k times), will contain k signatures that belong to the owners of the ticket. As we have explained in 3.2.5, if all the signatures checks are correct (the verification is made using the library XMLDSIG), then the web service generates a random challenge and sends it to the user. The user uses the challenge to create a signature. Next, the web service can verify the authenticity of the user. If not, the process stops automatically and sends a notification to refuse the user request.

5.2.4 Ticket Consumer

The application *eTickets* is one of the main actors of the developed system, which allows the user to register for the service, obtain, manage and transfer tickets and eventually consume them. To achieve persistence, we have used the SQLite language supported by Java in the Android platform. The *eTickets* application has two tables:

- *Credentials*: It stores the different credentials that a user has to use in the program. The user's credentials have a public key (*PublicKey*) and a private key (*PrivateKey*). These credentials also have the public key certificate issued by the external certification body.
- *Tickets*: It stores the tickets of the user. It has the following attributes:

- *ID* (integer): It consists of the ticket ID number (serial number).
- *Content* (text): Content ticket in text format.
- *SignerParameters* (array of bytes): It is the signature parameters used to sign the corresponding ticket.

The application has a menu (SlideMenuClickListener), which manages the different functions of the application. The available options are:

- *Home*: It gives access to presentation view. The customer cannot make any special functionality because it is the first view that he/she sees when turning on the mobile app.
- *Register*: It gives access to log view. It allows the user to register in the e-ticketing system.
- *eTickets*: It launches the view with the tickets list. Allows the user to see the tickets that owns and to get new ones.
- *Ticket Transfer*: This is a view to transfer a ticket to another user.
- *Ticket Spending*: It gives access to the view with the option to spend the tickets.

Finally, note that the implemented system uses three different communications technologies:

- *Internet*: Entities involved in the scheme communicate via HTTP or HTTPS connection.
- *NFC*: A sender can send an NFC message to a receiver without waiting for an answer.
- *Bluetooth*: Two entities establish a connection using Bluetooth technology and exchange data in a bidirectional channel.

To implement the Transfer Ticket operation, we use two wireless technologies such as NFC (unidirectional communication due to some constraints of the Android library) and Bluetooth (bidirectional communication). To combine them, we use *EasyNFC* library. NFC is used to establish the Bluetooth communication channel between sender and receiver of the e-ticket.

The issuer builds an NFC message using NFC Data Exchange Format (NDEF). The message is divided into different *Records*, where we can place the exchanged data with the format of string of bytes. In our case, in this message we put the essential information to establish the connection, which includes the physical address of the sending device and the communication channel.

The *eTicket* application includes some information so that the receiving device starts up the application as soon as it receives the notification. It also enables the Bluetooth of the device. Then, the application uses *EasyNFC* to open an internal Bluetooth connection server, which is awaiting connections from other devices. Once the transfer request is received, the app shows a menu to accept or cancel the ticket transfer (figure 3). If the user accepts it, it starts the ticket information exchange.

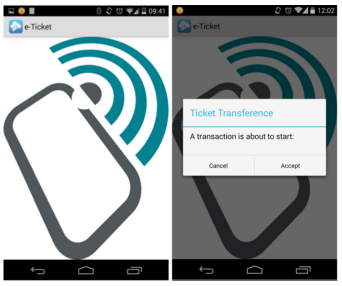


Fig. 3 e-Ticket Transfer: sender (left) and recipient (right)

5.3 Group signatures Implementation

This version of the e-ticketing system has the same features as the previous scheme with RSA signatures. The distinction from one to the other is the use of different cryptographic schemes. Thus, in this section, we only focus on the implementation of the Group Signature Cryptography (the rest of the features are shared with the RSA implementation).

In this case, we use the JPBC Java (Java Pairing Based Cryptography) library [14], which has all classes and procedures to deal with additive cyclic groups of prime order and bilinear pairing functions [15]. The bilinear pairing function can be set in different options, depending on the type of elliptic curve generated [14]. Due to efficiency reasons, we have chosen a type A curve [16]. Next, we will briefly review the Java classes we have created to implement the main operation of the e-ticketing scheme.

5.3.1 GroupManager Class

GroupManager class is responsible for generating all user credentials. This class performs the $KeyGen_G(n)$ operation with the n parameter, which represents the number of SDH tuples (A_i, x_i) (see 2.1), which are the users' private keys of a group.

In the following steps of the protocol, we also generate some random values $h \xleftarrow{R} G_1 \backslash \{1_{G_1}\}$ and two values $gmsk = (\xi_1, \xi_2)$, which are the secret master key of the group. These two values (ξ_1, ξ_2) are stored in a *GroupMasterSecretKey* class because they play an essential role: if a user commits fraud, $Open_G(gpk, gmsk, M, \sigma)$ operation is used to reveal the user identity.

Regarding the generation of credentials (SDH tuples), each x_i value is randomly selected from $\mathbb{Z}_p^*$, and it is associated to $A_i \leftarrow g_1^{1/(\gamma+x_i)}$. Each pair is stored in a different $SecretKey$ object, and it is the private key for each user in the group. Finally, it creates an object that represents the public key of the group ($GroupPublicKey$ class), where all public values are stored. Note that each user group will have a different $GroupPublicKey$ and $SecretKey$ objects.

5.3.2 Signer Class

This class aims to generate signatures according to the proposed group signature scheme. There are two signature types: regular and linkable. Each one is generated by a different function:$Sign_G(gpk, gsk[i], M)$ and $SignLinkable_G(gpk, gsk[i], M', \sigma, \alpha, \beta)$.

$Sign_G()$ operation receives as parameter a public key group $GroupPublicKey$ and a private key $SecretKey$, together with the message (in text form) to be signed. The result of the signature operation is the message and the $Sigma$ object which contains the values (T_1, T_2, T_3), the challenge, c, and the values $(s_\alpha, s_\beta, s_x, s_{\delta_1}, s_{\delta_2})$. Both elements (message and $Sigma$) are encapsulated in a $Signature$ class object, which is more flexible and follows the object-oriented paradigm.

Moreover, the $SignLinkable_G()$ operation is different from the conventional group signature. It operates over the $GroupPublicKey$ and $SecretKey$ parameters of the user, a message M' (text format), as well as a signature previously calculated, and the parameters α, β used in the previous computed signature. The output of the operation is structurally identical to the $Sign_G()$ operation: a $Signature$ object, the M' message and the σ' object.

5.3.3 Verifier Class

This class focuses primarily on verifying group signatures, whether regular or linked. Depending on the case, it executes the operation $Verify_G(gpk, M, \sigma)$ or $VerifyLinkable_G(\sigma, \sigma')$, respectively.

$Verify_G()$ function receives as a parameter a $GroupPublicKey$ and the $Signature$ object to be verified. Then, it checks whether the group signature $\sigma = (T_1, T_2, T_3, c, s_\alpha, s_\beta, s_x, s_{\delta_1}, s_{\delta_2})$ is valid or not. The signature will be valid only if the challenge c provided by the parameter matches with the new challenge c', calculated according to the protocol specification. The $VerifyLinkable_G()$ operation verifies the linkability of two $Signature$ objects. Thus, it checks that a user has generated both signatures. The operation verifies that the values (T_1, T_2, T_3) from the first signature match the ones from the second signature.

5.3.4 Zero-Knowledge-Proof (ZKP)

The purpose of ZKP is to verify that a signed message is being used for its proper owner. This is equivalent to saying that the user knows the secret parameters $(\alpha, \beta, x, \delta_1, \delta_2)$ which the group signature was generated with. To do that, we have implemented the following protocols: $ZKP_GCommit(M^*)$, $ZKP_GVerify(m', c', s')$ and $ZKP_GResponse(m', c')$.

$ZKPGResponse()$ receives, as a parameter, the object $ZKP_GCommit(M^*)$, which represents the commitment generated by the procedure with the same name, and a new challenge c'. The user is responsible for generating a response to the ZKP by computing the values $s' = ({s_\alpha}', {s_\beta}', {s_x}', {s_{\delta_1}}', {s_{\delta_2}}')$ with the appropriate algorithm.

Finally, given a $ZKPCommit$ commitment and a $ZKPResponse$ response generated by the user, the verifier performs the operation $ZKPGVerify()$, which verifies that the parameters of both operations match.

If these comparisons have a positive result, the function returns a boolean true value which confirms that the ZKP is correct. Otherwise, it returns false.

5.3.5 Final Encapsulation

We have wrapped all implemented classes together as a single unit. Therefore, all the operations of the cryptographic protocol have been encapsulated within a single Java library. So, this new library can be imported by any external entity, without having access to the implemented internal code.

Thus, the implemented external entities of this version of the e-ticketing scheme have the same features than the entities in the RSA implementation, but now with the capability of executing the above cryptographic operations.

5.4 Cost results

We have studied the cost of execution of both protocols as well as the partial times of each phase of the protocols (*Registration*, *Ticket Issue*, *Ticket Transfer* and *Ticket Verification*), in order to identify the most costly parts of each protocol in the client (user) side.

We have executed a set of experimets using four user agents (four smartphones) for each measure. The devices are a Samsung Galaxy S3, a Google Nexus 5, a Samsung Galaxy Fame and a Google Nexus S.

In sections 5.4.1 and 5.4.2 we present the average values of the phases of both proposals. For the second proposal, the group signature-based one, some operations present large delays and for this reason we have broken down the operations of each phase (section 5.4.3) in order to detect which ones are the main cause of the delays.

5.4.1 Time cost results for the RSA-based proposal.

In this section we present the cost results for the RSA-based proposal. Figure 4 shows the average time (in seconds) required to complete each transaction (*Registration*, *Ticket Issue*, *Ticket transfer* and *Ticket Verification* phases) taking into account the interaction with the other entities. We detail the results obtained for the four user agents: Samsung Galaxy S3, Google Nexus 5, Samsung Galaxy Fame and

Google Nexus S. In general, all the transactions are considered practical because the computational cost is lower than 5 seconds in all operations required for the use of a non-transferred ticket. As expected, the ticket transfer is the most costly operation.

Quadrant and Antutu are benchmarks devoted to evaluate the CPU and graphics performance of electronic devices. They also assess the memory speed and the external and internal storage read/write speeds. The final score received after a running of a benchmark test is a combination of the evaluated features. According to these benchmarks [6], Nexus 5 and Galaxy S3 have a better performance results than Galaxy Fame and Nexus S. The performance evaluations of our protocol follow the expected relationship according to the scores received by the devices when they are evaluated by the previous mentioned benchmarks. Thus, Nexus 5 and Galaxy S3 run our protocol faster than Galaxy Fame and Nexus S.

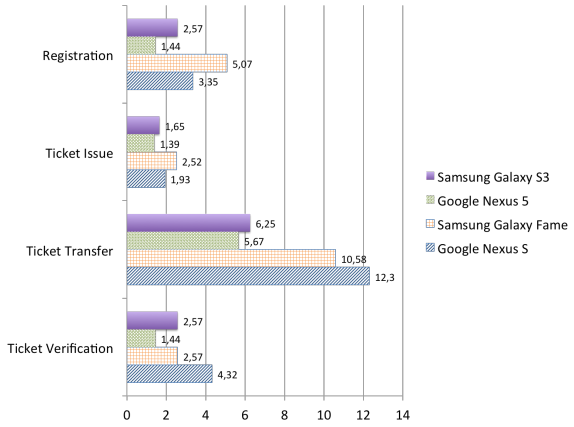


Fig. 4 Execution time of the phases of the RSA Signature Proposal.

5.4.2 Time cost results for the Group Signature-based proposal

The results for the Group Signature-based proposal presented in this section will be compared with the results of the RSA-based proposal. Figure 5 shows the average time (in seconds) required to complete each transaction (*Registration*, *Ticket Issue*, *Ticket transfer* and *Ticket Verification* phases). As expected, due to the complexity of the Group Signature-based proposal, here we have greater delays compared with the RSA-based proposal. Once again, the ticket transfer is the most costly phase. In this phase we can observe a great difference among the most powerful devices (Samsung Galaxy S3 and Google Nexus 5) and the others. For these devices the required time to perform a ticket transfer is around four times the required time for the RSA-based proposal. Taking into account these results, the execution cost (time) for the Group signature-based proposal is above the desired performance parameters for the protocol (even if the ticket is used without being transferred). The differences among the devices show that the evolution in the computational power of the smartphones will lead, eventually, to a better performance, allowing in the near future the use of complex protocols, as the group signature-based protocol. The devices with better scores in the benchmarks (Nexus 5 and Galaxy S3) can perform the most costly operation with half as time as less powerful devices (Galaxy Fame and Nexus S).

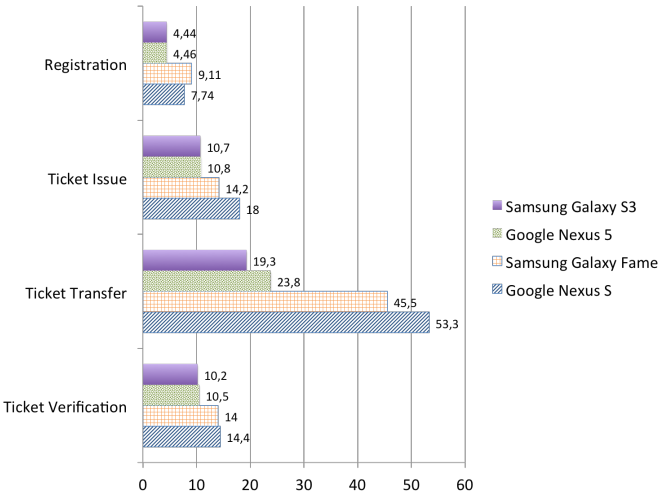


Fig. 5 Execution Time of the phases of the Group Signature Proposal.

5.4.3 Distribution of the cost among operation for the Group Signature-based proposal

While the cost results for the RSA-based proposal are low and make the protocol viable (in the sense that all the operation can be completed in less than 5 seconds for a non-transferred ticked), the cost results for the Group Signature-based proposal are, in some cases, too high. Due to the fact that the Group Signature-based proposal offers better anonymity (strong anonymity versus weak anonymity), we want to discover which of the operations involved in the Group Signature-based proposal are the cause of greater

6
http://www.hardwarezone.com.sg/review-google-nexus-s-succeeding-one/performance-part-i-8/
http://www.verygoodreview.com/android-phone/samsung-galaxy-fame-gt-s6810-review.html/
http://www.verygoodreview.com/comparison/samsung-galaxy-s-iii-s3-gt-i9300-vs-htc-one-x.html/
http://mobileandphone.com/lg-google-nexus-5-vs-samsung-galaxy-s4-vs-lg-g2/

computational cost. In this section, we have itemized the operations of each phase of the protocol and we have evaluated their cost. The operations are described in Table 1 and match with the steps of the protocol description in section 3.3.

REGISTRATION	ISSUE
WS (Web Service) Connection Key Charge in the Smartphone Smartphone Configuration	WS Request and Nonce Return Signature Creation Ticket Emission by WS XMLDSig Verification
TRANSFER (A to B)	**VERIFICATION**
ZKP Commit (A) Bluetooth Transfer (A) Ticket XMLDsig Verification (B) Ticket Signature Verification (B) Challenge generator (B) ZKP Response (A) ZKP Verification (B) New signature generation (A) Receptor Signature Verif. (A) New Linkable Signature Verif. (A) Simple Verification (B) Linkable Verification (B)	Connect, Send Ticket to server WS Verif. and Nonce creation Linkable Signature Creation Linkable signature Verification

Table 1 Operations for each phase.

For each phase of the protocol, we have calculated the cost and we have detailed it for the operations included in Table 1. We represent the costs in Figures 6, 7, 8 and 9. This way, it can be easily observed which operations contribute mainly to the total cost.

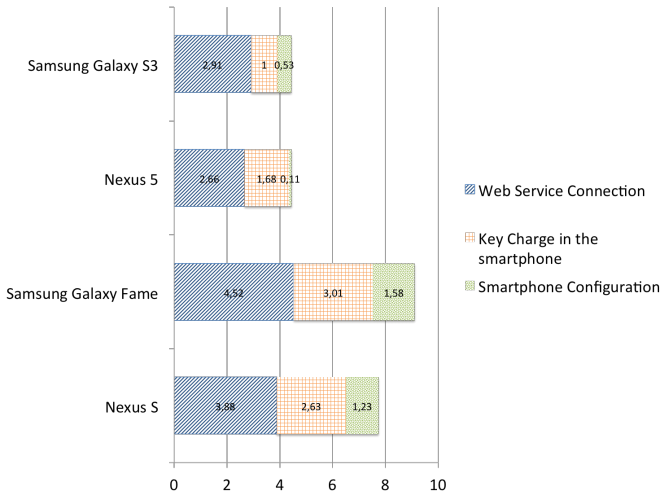


Fig. 6 Registration phase operations in the Group Signature Proposal.

In Figure 6, we can see that the most costly operation for both smartphones is the connection to the web service. In this operation, the user sends her data to the web service and obtains the keys. This stage is not critical for the total delay since the registration is not executed every time the user wants to use a ticket.

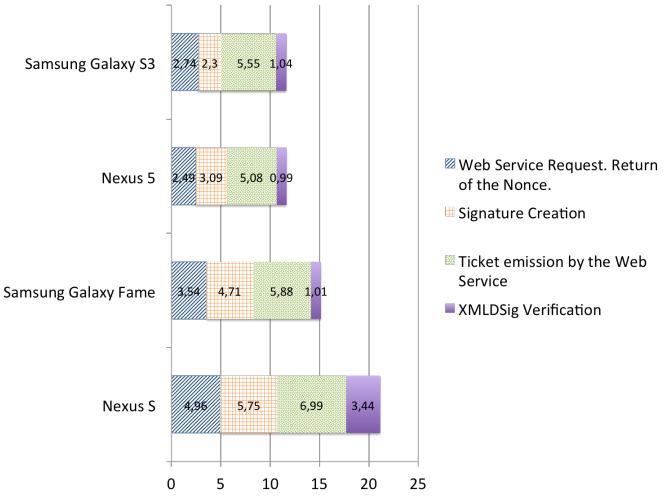


Fig. 7 Issue phase operations in the Group Signature Proposal.

In Figure 7, we observe that the generation of the ticket is the part with the highest cost. At the same time, we can see that the verification of the XML signature is very quick for all the devices while the time used for the generation of the signature differs from one device to the other.

The most complex phase is the transfer of the ticket. Although it is not executed for each ticket, the time required for the transfer will have to be reduced in a viable implementation. The results of the performance evaluation, see Figure 8, gives us the knowledge of which operations are more costly. This information will be very useful in the design of an improved version of the protocol. The time required for some opertions, like the Bluetooth transfer and the creation and verification of signatures are reduced considerably in the executions using the most powerful devices.

Finally, for the verification of the ticket, the generation of the linkable signature and the connection with the server are the most costly operations, while the verification is very quick, as it can be seen in Figure 9. However, for this stage, the delay is more balanced among the operations, for all the devices.

6 Comparison of the approaches and Conclusions

We have presented two approaches to obtain anonymity in transferable electronic ticketing systems. These approaches guarantee the anonymity for their users even when the system allows the transfer of tickets between them through payment or loan. We have to take into account that the kind of attacks that can be performed in a transferable system are

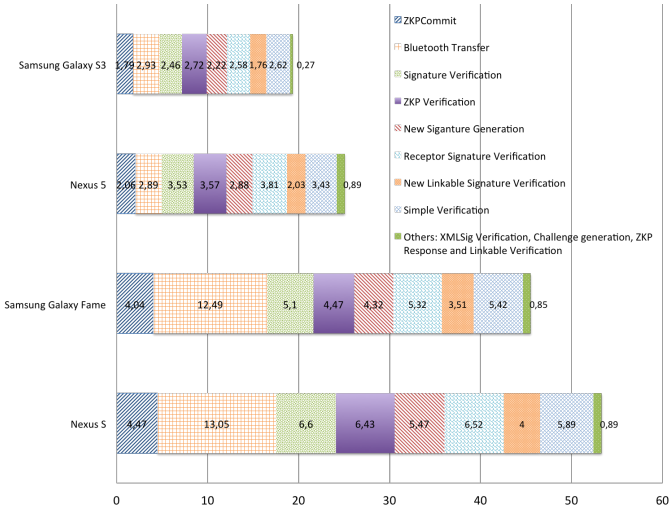


Fig. 8 Transference phase operations in the Group Signature Proposal.

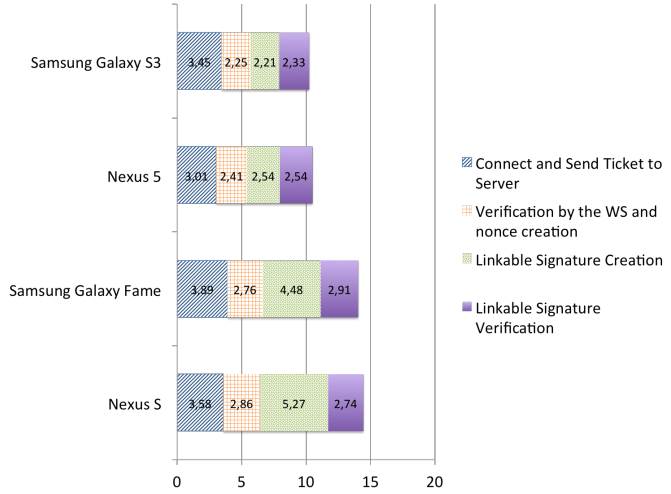


Fig. 9 Verification phase operations in the Group Signature Proposal.

greater than those in a non-transferable system due to the possibility of attack by confabulated users.

The first approach provides anonymity using RSA signatures and a registration authority. A transfer commitment chain is generated to maintain anonymity as well as security during the transfers. Users are not identified if they use the system properly. Moreover, the system allows the revocation of the identity of fraudulent users. However, if the user uses her credentials several times, the tickets can be linked. For these reason, the anonymity provided by this approach is called weak anonymity.

The second scheme proposed is fully anonymous because a group signature scheme has been used. It allows the issuer to verify that the user belongs to a valid group of users, yet it cannot identify which one she is. If the user tries to commit fraud, the group manager can revoke her anonymity. To achieve transferability, we use a linkable group signature scheme. With this technique, group signatures from the users involved in the transfer operation are used in order to generate a ticket transfer agreement, which could be further used as evidence in case of conflict between the parties.

This paper includes the development and evaluation of the performance of the protocols in a mobile platform, in order to check its feasibility. We have described the implementation of both proposals including requirements, RSA implementation and group signature implementation.

The performance of the system will be a very important point because users have to be able to obtain, validate or transfer the ticket in a short time. For this reason we have evaluated both proposals, with different degrees of anonymity. The second approach, more robust, has presented efficiency problems to be used in some current devices and, although the increased complexity of the second approach could be justified in order to obtain a greater degree of anonymity, its viability depends strongly on the development of future devices and networks (communication cost are an important part of the total cost), in which the cost of execution will surely be reduced. Moreover, the results of the performance evaluation give us the knowledge of which operations are the most costly. This information will be very useful in order to design an improved version of the protocol with the aim of obtaining an efficient fully anonymous transferable e-ticketing scheme.

ACKNOWLEDGEMENTS

This work was partially supported by the Spanish Government under CO-PRIVACY TIN2011-27076-C03-01, AccessTur TIN2014-54945-R, SmartGlacis TIN2014-57364-C2-R and MobileKey RTC-2014-2552-7 projects. Some of the authors are members of the UNESCO Chair in Data Privacy, yet the views expressed in this paper neither necessarily reflect the position of the UNESCO nor commit with that organization.

References

1. A. Vives-Guasch and M. Payeras-Capellà and M. Mut Puigserver and Jordi Castellà-Roca and J. Ferrer-Gomila, ”Anonymous and Transferable Electronic Ticketing Scheme”, En Data Privacy Management and Autonomous Spontaneous Security, DPM 2013, LNCS 8247, Pages 100–113, Springer 2013.
2. Blazy, Olivier and Canard, Sébastien and Fuchsbauer, Georg and Gouget, Aline and Sibert, Hervé and Traoré, Jacques, ”Achieving Optimal Anonymity in Transferable E-Cash with a Judge”, In Progress in Cryptology, AFRICACRYPT 2011, volume 6737 of Lecture Notes in Computer Science, pages 206-223. Springer Berlin Heidelberg, 2011.
3. D. Boneh, X. Boyen and H. Shacham, ”Short group signatures”, In CRYPTO, volume 3152 of LNCS, pages 41-55. Springer, 2004.

4. Y. Chen, C.-L. Chen, and J.-K. Jan, "A mobile ticket system based on personal trusted device". Wireless Personal Communications, 40(4):569-578, 2007.
5. S. Ghiron, S. Sposato, C. Medaglia, and A. Moroni. "Nfc ticketing: A prototype and usability test of an nfc-based virtual ticketing application". In Near Field Communication, NFC '09, p. 45-50. IEEE, 2009.
6. T. S. Heydt-Benjamin, H.-J. Chae, B. Defend, and K. Fu. "Privacy for public transportation". In 6th Workshop on Privacy Enhancing Technologies (PET 2006), LNCS 4258, p. 1-19. Springer, 2006.
7. A. P. Isern-Deya, Vives-Guasch, M. Mut-Puigserver, M. Payeras-Capella and J. Castella-Roca. "A secure automatic fare collection system for time-based or distance-based services with revocable anonymity for users". The Computer Journal, 2012.
8. D. Jao and K. Yoshida. "Boneh-boyen signatures and the strong difie-hellman problem". In Pairing Based Cryptography, volume 5671 of LNCS, pages 41-55. Springer, August 2009.
9. O. Jorns, O. Jung, and G. Quirchmayr. "A privacy enhancing service architecture for ticket-based mobile applications". In Availability, Reliability and Security, pages 374-383, 2007. ARES 2007 - The International Dependability Conference. vol. 24.
10. M. Mut-Puigserver, M. M. Payeras-Capella, J.-L. Ferrer-Gomila, A. Vives-Guasch, and J. Castella-Roca. "A survey of electronic ticketing applied to transport. Computers and Security, 31(8):925-939, 2012.
11. D. Quercia and S. Hailes. "Motet: Mobile transactions using electronic tickets". In Security and Privacy for Emerging Areas in Communications Networks, Proceed- ings, pages 374-383, 2005. vol. 24.
12. A. Vives-Guasch, J. Castella-Roca, M. Payeras-Capela, and M. Mut. "An electronic and secure automatic fare collection system with revocable anonymity for users". In Advances in Mobile Computing and Multimedia (MoMM), 2010.
13. A. Vives-Guasch, M. M. Payeras-Capella, 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 Vol.E95-D No.1, 2012.
14. Angelo De Caro and Vincenzo Iovino. "jPBC: Java pairing based cryptography". Proceedings of the 16th IEEE Symposium on Computers and Communications, ISCC 2011. pp.850-855. IEEE 2011.
15. Alfred Menezes. "An introduction to pairing-based cryptography". Recent Trends in Cryptography. Contemporary Mathematics. Volume 477. American Mathematical Society. 2009
16. Applied Cryptography Group, "PBC library manual". 2006. Available at url: https://crypto.stanford.edu/pbc/manual.