E-Ticketing Scheme for Mobile Devices with Exculpability

Arnau Vives-Guasch[1], Magdalena Payeras-Capella[2], Macià Mut-Puigserver[2], and Jordi Castellà-Roca[1]

[1] Dpt. d'Enginyeria Informàtica i Matemàtiques, UNESCO Chair in Data Privacy, Universitat Rovira i Virgili, Av. Països Catalans 26, E-43007 Tarragona, Spain
{arnau.vives,jordi.castella}@urv.cat

[2] Dpt. de Ciències Matemàtiques i Informàtica, Universitat de les Illes Balears, Ctra. de Valldemossa, km 7,5. 07120 Palma de Mallorca, Spain
{mpayeras,macia.mut}@uib.es

Abstract. An electronic ticket is a contract, in digital format, between the user and the service provider, and reduces both economic costs and time in many services such as air travel industries or public transport. However, the electronic ticket security has to be strongly guaranteed, as well as the privacy of their users. We present an electronic ticketing system that considers these security requirements and includes the exculpability as a security requirement for these systems, i.e users and the service provider can not falsely accuse each other of misbehaviour. The system ensures that either both parties receive their desired data from other or neither does (fair exchange). Furthermore, this scheme takes special care of the computational requirements on the users' side, as we consider the use of mobile devices with light-weight cryptography, because they are the best positioned in order to manage the electronic tickets in the near future.

Keywords: e-ticketing, e-commerce, security, privacy, exculpability.

1 Introduction

Information technologies (IT) are becoming usual in our society as they progressively replace the use of paper in many of our common operations. An example of paper ticket could be the air flight boarding pass. Vodafone and Spanair[1] conducted an e-ticketing test in Spain (May 2007). The passengers received the electronic boarding pass in their mobile phone, and they were able to go directly to the security control area, and later board. The International Air Transport Association (IATA) started in 2004 a program to introduce the use of electronic tickets. IATA estimates that the no-use of paper tickets will reduce the costs by US$ 3000M[2], boosting disintermediation by using electronic tickets. The electronic tickets have not been used only as a boarding pass. They can also be used

[1] http://www.spanair.com/web/es-es/Sobre-Spanair/Noticias-y-eventos/Spanair-y-Vodafone-Espana-presentan-la-tarjeta-de-embarque-movil/

[2] IATA:http://www.iata.org/pressroom/pr/2008-31-05-01.htm

J. Garcia-Alfaro et al. (Eds.): DPM 2010 and SETOP 2010, LNCS 6514, pp. 79–92, 2011.
© Springer-Verlag Berlin Heidelberg 2011

in multiple transport services. The AMSBUS[3] booking system from the Czech Republic allows the purchase of SMS tickets. The passenger receives the ticket in her mobile phone; then, the user shows the message to the ticket inspector when needed. Leeds United[4] supporters can book one of their sport events and later receive an SMS with the booking confirmation together with some added information such as the assigned seat.

These examples show the progressive introduction of electronic tickets on different kinds of services and the increasing use of mobile phones in all of them as the most suitable e-ticket storage device. In addition to that, the real application of these electronic ticketing systems depends on their security, due to the ease of copy of electronic data. Electronic tickets have to keep the same security that is offered in paper tickets.

1.1 Organization

An analysis of the previous works related to our proposal is presented in Section 2. Later, in Section 3, our scheme is accurately defined. The security and privacy analysis of the scheme is detailed in Section 4. Finally, the conclusions and future work are described in Section 5.

2 Previous Works

First of all, in Section 2.1, we start presenting the security requirements that have to be achieved in these systems, as well as we introduce a new security requirement that is not achieved in the previous works and which could be taken into account in the future: *exculpability*. Once the security requirements are described, the e-ticketing proposals have been classified in Section 2.2.

2.1 Security Requirements

The e-ticketing systems have to consider and guarantee the following security requirements:

- Authenticity : e-tickets should take measures to avoid falsification.
- Non-repudiation: the issuer can not deny the emission of an authentic ticket.
- Integrity: an issued ticket can not be further modified by anyone.
- Anonymity: there are two anonymity degrees: non-anonymous (the service requires user identification and authentication) and revocable anonymous (the service is anonymous, but it can be revoked if the user misbehaves).
- Reusability: a ticket could be used once (non-reusable) or many times (reusable). In both cases, ticket overspending has to be prevented.
- Expiry date: a ticket could be only valid during a time interval.
- Online/Offline: ticket verification can require a persistent connection with a trusted centralized system (online); otherwise, that connection is never necessary (offline).

[3] AMSBUS: `http://www.svt.cz/en/amsbus/`
[4] Leeds United: `http://www.leedsunited.com/`

Exculpability. None of the analyzed proposals deals with exculpability; that is, the service provider can not falsely accuse the user of ticket overspending, and the user is able to demonstrate that she has already validated the ticket before using it. The exculpability is an interesting issue in our case, because the e-ticketing scheme should ensure that either both parties (users and provider) receive their desired data (e-ticket and the validated e-ticket) from other or neither does (fair exchange). The parties agree to reveal its data only if the other part also agrees. If any party deviates from the scheme then it can be identified as the culprit by the Trusted Third Party (TTP). Our scheme defined in Section 3 takes exculpability as a security requirement for an e-ticketing system, as a first step to include this security requirement in future works.

2.2 Classification of Proposals

In this section, the proposals have been differentiated by the devices used in the systems: the *smart-card-based*, and the *non-smart-card-based* ones, with a deeper analysis performed in the non-smart-card-based proposals, taking into account their anonymity compliance.

Smart-card-based. Smart-card-based proposals [5,7,13,12,14,19,18] establish a communication channel with the verification system. Thus, the most sensitive operations are sent to the smart-card through this channel. The smart-card verifies each operation, so that users can not perform any non-allowed action. Security of smart-card-based systems rely on the smart-card security. If the smart-card security is compromised, the security of the entire system is also compromised. One recent example is the case of the Mifare cards, where in [10] the authors succeeded to compromise them. As a result, all the entire public transport system that used exclusively Mifare cards was compromised.

Non-smart-card-based. Non-smart-card-based systems [15,6,1,8,16,9,3] allow to perform applications with higher computation requirements while taking advantage of their high storage capacity and also their wireless short-range communication resources; this is the case of the mobile phones, smart phones or PDA's. However, as these devices are not considered tamper-proof devices, e-ticketing systems require then high-level cryptographic protection in order to assure that users follow the e-ticketing protocol correctly. We classify the non-smart-card-based systems depending on *non-anonymous* and *revocable-anonymous* compliance.

Non-Anonymous proposals. Some proposals are oriented to services where anonymity can not be provided to the user, or simply, these systems are not conceived to achieve anonymity at all. Between the proposals that do not consider anonymity, digital signatures are commonly used [6,1].

Revocable-Anonymous proposals. In the analyzed proposals, digital signatures are commonly used [15,8,16,9,3], providing ticket authenticity, non-repudiation

and integrity. Most of them are based on Chaum's blind signatures [2] with the use of pseudonyms for revocable anonymity. There is a diversity of considerations in the ticket reusability; Patel and Crowcroft [15] believe that tickets can be reusable; Haneberg et al. [8], Heydt-Benjamin et al. [9] and Chen et al. [3] do not consider ticket reusability; finally, Quercia and Hailes [16] consider that reusability mainly depends on the service. There is a remarkable equality between the proposals that use online verification [15,9] and the ones which use offline verification [8,16,3].

3 E-Ticketing Scheme

The scheme has the following actors: the user $\mathcal{U}$; the ticket issuer $\mathcal{I}$, who sends a valid ticket to $\mathcal{U}$; the service provider $\mathcal{P}$, who verifies the ticket and gives the service; and finally the TTP $\mathcal{T}$, who preserves $\mathcal{U}$'s anonymity, and gives a valid non-identity pseudonym to $\mathcal{U}$. The e-ticketing scheme has been designed for mobile devices, reducing the computation requirements in the user side, and providing the basic security requirements (authenticity, non-repudiation and integrity) together with expiry date, revocable anonymity and exculpability. The ticket verification is performed offline when there is only one provider for each service, although the scheme could be extended with multiple providers that offer the same service. In that extended case, the scheme would prevent overspending through online verification, requiring then the interconnection of the service providers which offer the same service. In any case, the connection with the issuer is never necessary. In Table 1, we define the details of our proposal, as well as some notation that is used in the scheme.

Table 1. Details of the proposal: security requirements, actors, ticket information and receipt information

Security Requirements			
Authenticity		Non-repudiation	
Integrity		Revocable anonymity	
Non-overspending		Offline verification	
Expiry date		Exculpability	
Actors			
User	$\mathcal{U}$	Service Provider	$\mathcal{P}$
Ticket Issuer	$\mathcal{I}$	Trusted Third Party	$\mathcal{T}$
Ticket Information (T)			
Serial number	Sn	Issuer	Is
Service	Sv	Terms and conditions	Tc
User pseudonym	$\mathsf{Pseu}_{\mathcal{U}}$	Attributes	At
Type of ticket	Ty	Encrypted verification data	$\delta_{\mathcal{T},\mathcal{P}}$
Validity time	Tv	Date of issue	Ti
Exculpability ($\mathcal{U}$)	$\mathsf{h}_{r_{\mathcal{U}}}$	Exculpability ($\mathcal{P}$)	$\mathsf{h}_{r_{\mathcal{I}}}$
Digital signature of $\mathcal{I}$	$\mathsf{Sign}_{\mathcal{I}}(\mathsf{T})$		
Receipt Information (R)			
Encrypted exculpability ($\mathcal{P}$)	$\mathsf{A}_{\mathcal{P}}$	Timestamp	τ_i
Ticket serial number	$\mathsf{T.Sn}$	Digital signature of $\mathcal{P}$	$\mathsf{Sign}_{\mathcal{P}}(\mathsf{R})$

3.1 Phases

The phases of our system are: *Pseudonym Renewal*, where the user obtains a new temporal pseudonym to be used in the system without linkage to user's identity (if user behaves correctly); *Ticket Purchase*, that consists on the payment of the service and reception of the ticket; and *Ticket Verification*, where the user shows the ticket to the service provider in order to be checked and validated. Other phases considered in the system are claims. These claims should only be executed in case of controversial situations during the *Ticket Verification* phase: *Claim* m_2 *Not Received* (when $\mathcal{U}$ sends the first step of the verification m_1 but does not receive m_2 by $\mathcal{P}$, or the information is not correct); *Claim* m_3 *Not Received* (when $\mathcal{P}$ sends the second step of the verification m_2 but does not receive m_3 by $\mathcal{U}$, or the information is not correct); and *Claim* m_4 *Not Received* (when $\mathcal{U}$ sends the third step of the verification m_3 but does not receive m_4 by $\mathcal{P}$, or the information is not correct). Users have a digital credential ($\mathsf{Cert}_{\mathcal{U}}$) for authentication to the TTP only, as the system is anonymous, and all further movements in the system are tracked only with the assigned temporal pseudonym ($\mathsf{Pseu}_{\mathcal{U}}$).

Pseudonym Renewal. The user $\mathcal{U}$ contacts the pseudonym manager $\mathcal{T}$ in order to renew the assigned pseudonym. The certificate $\mathsf{Cert}_{\mathcal{U}}$ identifies $\mathcal{U}$ through a secure connection established between the two parties. The system has the public parameters (α, p, q), where α is a generator of the group G with order p, being p and q large primes achieving $p = 2q + 1$. $\mathcal{U}$ generates a random value $\mathsf{x}_{\mathcal{U}} \overset{R}{\leftarrow} \mathbb{Z}_q$ and computes $\mathsf{y}_{\mathcal{U}} = \alpha^{\mathsf{x}_{\mathcal{U}}} \pmod p$ in order to receive a valid signed pseudonym $\mathsf{Pseu}_{\mathcal{U}}$ from $\mathcal{T}$. $\mathcal{U}$ and $\mathcal{T}$ have their own pair of keys used for signature and encryption of the transmitted data between them.

authenticateUser User $\mathcal{U}$ follows the next steps:

1. generates $\mathsf{x}_{\mathcal{U}} \overset{R}{\leftarrow} \mathbb{Z}_q$, and computes $\mathsf{y}_{\mathcal{U}} = \alpha^{\mathsf{x}_{\mathcal{U}}} \pmod p$;
2. computes the signature $\mathsf{sk}_{\mathcal{U}}(\mathsf{h}_{\mathsf{y}_{\mathcal{U}}})$ [5] where $\mathsf{h}_{\mathsf{y}_{\mathcal{U}}} = hash(\mathsf{y}_{\mathcal{U}})$ [6];
3. encrypts $\mathsf{y}_{\mathcal{U}}$ with the $\mathcal{T}$'s public key: $\mathsf{pk}_{\mathcal{T}}(\mathsf{y}_{\mathcal{U}})$ [7];
4. sends $(\mathsf{sk}_{\mathcal{U}}(\mathsf{h}_{\mathsf{y}_{\mathcal{U}}}), \mathsf{Cert}_{\mathcal{U}}, \mathsf{pk}_{\mathcal{T}}(\mathsf{y}_{\mathcal{U}}))$ to $\mathcal{T}$;

generatePseudonym Pseudonym Manager $\mathcal{T}$ executes:

1. decrypts $\mathsf{sk}_{\mathcal{T}}(\mathsf{pk}_{\mathcal{T}}(\mathsf{y}_{\mathcal{U}})) \rightarrow (\mathsf{y}_{\mathcal{U}})$;
2. verifies $\mathsf{y}_{\mathcal{U}}$: $\mathsf{pk}_{\mathcal{U}}(\mathsf{sk}_{\mathcal{U}}(\mathsf{h}_{\mathsf{y}_{\mathcal{U}}})) \rightarrow (\mathsf{h}_{\mathsf{y}_{\mathcal{U}}}) \overset{?}{=} hash(\mathsf{y}_{\mathcal{U}})$;
3. if correct, then computes the signature of $\mathsf{sk}_{\mathcal{T}}(\mathsf{h}_{\mathsf{y}_{\mathcal{U}}})$; and
4. sends $\mathsf{Pseu}_{\mathcal{U}} = (\mathsf{y}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{T}}(\mathsf{h}_{\mathsf{y}_{\mathcal{U}}}))$ to $\mathcal{U}$.

verifyPseudonym $\mathcal{U}$ computes:

1. verifies $\mathsf{y}_{\mathcal{U}}$: $\mathsf{pk}_{\mathcal{T}}(\mathsf{sk}_{\mathcal{T}}(\mathsf{h}_{\mathsf{y}_{\mathcal{U}}})) \rightarrow (\mathsf{h}_{\mathsf{y}_{\mathcal{U}}}) \overset{?}{=} hash(\mathsf{y}_{\mathcal{U}})$;

[5] Note that $\mathsf{sk}_{\mathcal{E}}(content)$ means the decryption of *content* or the generation of a signature with its content *content* by using the private key of the entity $\mathcal{E}$.

[6] Note that $hash()$ is a public cryptographic one-way summarizing function that achieves collision-resistance.

[7] Note that $\mathsf{pk}_{\mathcal{E}}(content)$ means the encryption of *content* or the verification of a signature *content* by using the public key of the entity $\mathcal{E}$.

Ticket Purchase. The user establishes a connection with the ticket issuer $\mathcal{I}$ in order to receive the ticket. This connection could be established through an anonymous channel like TOR [4], guaranteeing then user's privacy. There are current contributions [8] that have implemented TOR for mobile devices with Android. $\mathcal{I}$ has a key pair and its public key certificate ($\mathsf{Cert}_{\mathcal{I}}$). Users do not use their personal keys (it would cause loss of anonymity); they use the temporal pseudonyms and authenticate through the Schnorr's Zero-Knowledge Proof (ZKP) [17]. The payment method is considered as out of scope in this proposal as we focus on the privacy given to user when joining/exiting the system, and using the service.

$\mathcal{I}$ generates the ticket with the information and its digital signature, together with the secret value $\mathsf{r}_{\mathcal{I}}$ and the secret shared key (they are decryptable only by $\mathcal{P}$ and $\mathcal{T}$) in order to let the provider show the secret value $\mathsf{r}_{\mathcal{I}}$ later, in the verification phase. The ticket issuer $\mathcal{I}$ and the user $\mathcal{U}$ follow this protocol:

getService $\mathcal{U}$ executes:

1. selects and pays for the desired service Sv;
2. generates a random value $\mathsf{r}_{\mathcal{U}} \overset{R}{\leftarrow} \mathbb{Z}_q$, and computes $\mathsf{h}_{\mathsf{r}_{\mathcal{U}}} = hash(\mathsf{r}_{\mathcal{U}})$;
3. computes $\mathsf{H}_{\mathcal{U}} = \alpha^{r_{\mathcal{U}}} \pmod p$;
4. generates two more random values $a_1, a_2 \overset{R}{\leftarrow} \mathbb{Z}_q$ to be used in the Schnorr proof;
5. computes $A_1 = \alpha^{a_1} \pmod p$;
6. computes $A_2 = \alpha^{a_2} \pmod p$;
7. sends $(\mathsf{Pseu}_{\mathcal{U}}, \mathsf{H}_{\mathcal{U}}, A_1, A_2, \mathsf{h}_{\mathsf{r}_{\mathcal{U}}}, \mathsf{Sv})$ to the ticket issuer $\mathcal{I}$.

getChallenge $\mathcal{I}$ follows the next steps:

1. generates and sends a challenge $\mathsf{c} \overset{R}{\leftarrow} \mathbb{Z}_q$ for $\mathcal{U}$;
2. asynchronously, for optimization, pre-computes $\mathsf{y}_{\mathcal{U}}{}^{\mathsf{c}} \pmod p$;
3. asynchronously, for optimization, pre-computes $\mathsf{H}_{\mathcal{U}}{}^{\mathsf{c}} \pmod p$;

solveChallenge $\mathcal{U}$ computes:

1. computes $\mathsf{w}_1 = a_1 + \mathsf{c} \cdot \mathsf{x}_{\mathcal{U}} \pmod q$;
2. computes $\mathsf{w}_2 = a_2 + \mathsf{c} \cdot \mathsf{r}_{\mathcal{U}} \pmod q$;
3. encrypts $(\mathsf{w}_1, \mathsf{w}_2)$ and sends it to $\mathcal{I}$: $\mathsf{pk}_{\mathcal{I}}((\mathsf{w}_1, \mathsf{w}_2))$;
4. pre-computes the shared session key used in the ticket verification: $\mathsf{K} = hash(\mathsf{w}_2)$;

getTicket $\mathcal{I}$ follows the next steps:

1. decrypts $\mathsf{sk}_{\mathcal{I}}(\mathsf{pk}_{\mathcal{I}}(\mathsf{w}_1, \mathsf{w}_2)) \rightarrow (\mathsf{w}_1, \mathsf{w}_2)$;
2. computes $\alpha^{\mathsf{w}_1} \pmod p$;
3. computes $\alpha^{\mathsf{w}_2} \pmod p$;
4. verifies $\alpha^{\mathsf{w}_1} \overset{?}{=} A_1 \cdot \mathsf{y}_{\mathcal{U}}{}^{\mathsf{c}} \pmod p$;
5. verifies $\alpha^{\mathsf{w}_2} \overset{?}{=} A_2 \cdot \mathsf{H}_{\mathcal{U}}{}^{\mathsf{c}} \pmod p$;
6. computes the shared session key: $\mathsf{K} = hash(\mathsf{w}_2)$;
7. obtains a unique serial number Sn, and a random value $\mathsf{r}_{\mathcal{I}} \overset{R}{\leftarrow} \mathbb{Z}_p$;
8. computes $\mathsf{h}_{\mathsf{r}_{\mathcal{I}}} = hash(\mathsf{r}_{\mathcal{I}})$;
9. composes $\kappa = (\mathsf{K}, \mathsf{r}_{\mathcal{I}})$ and signs it $\kappa^* = (\kappa, \mathsf{Sign}_{\mathcal{I}}(\kappa))$;
10. encrypts κ^* with a digital envelope which is decryptable by the TTP $\mathcal{T}$ and the provider $\mathcal{P}$ for possible future controversial situations during the ticket verification: $\delta_{\mathcal{T},\mathcal{P}} = \mathsf{pk}_{\mathcal{T},\mathcal{P}}(\kappa^*)$. If $\mathcal{I}$ tries to falsify $\delta_{\mathcal{T},\mathcal{P}}$, the $\mathcal{T}$ can check that information and, demonstrate that $\mathcal{I}$ is the culprit;

[8] `http://sourceforge.net/apps/trac/silvertunnel/wiki/TorJavaOverview`

11. fills out the ticket information T (Sn, Sv, $\mathsf{Pseu}_{\mathcal{U}}$, Tv, Ti, $\mathsf{h}_{r_{\mathcal{I}}}$, $\mathsf{h}_{r_{\mathcal{U}}}$, $\delta_{\mathcal{T},\mathcal{P}}$, etc.);
12. digitally signs the ticket T, and obtains the signed ticket, $\mathsf{Sign}_{\mathcal{I}}(\mathsf{T}) = \mathsf{sk}_{\mathcal{I}}(hash(\mathsf{T}))$, and $\mathsf{T}^* = (\mathsf{T}, \mathsf{Sign}_{\mathcal{I}}(\mathsf{T}))$;
13. sends T^* to the user $\mathcal{U}$.

receiveTicket $\mathcal{U}$ executes:

1. verifies the digital signature $\mathsf{Sign}_{\mathcal{I}}(\mathsf{T})$ of the ticket T using the issuer's certificate;
2. verifies that ticket T data and the performed request match;
3. verifies the ticket validity ($\mathsf{T.Ti}, \mathsf{T.Tv}$);
4. verifies $\mathsf{T.Pseu}_{\mathcal{U}}$;
5. stores $(\mathsf{T}^*, \mathsf{r}_{\mathcal{U}})$ in the device.

Ticket Verification. When the user wants to use the service, she must verify the ticket in advance. For simplicity, we present the ticket verification with only one provider, so the service provider $\mathcal{P}$ never needs permanent communication with the ticket issuer. Nonetheless, the protocol can be extended for multiple providers. In that case, all the service providers should be connected to a central repository of spent tickets in order to control ticket overspending. The user only interacts with the service provider, but in controversial situations, she and/or the service provider could interact directly with the TTP through a resilient connection in order to preserve the security requirements of the protocol. If user misbehaved, her identity could be revoked, enabling to take further actions.

$\mathcal{U}$ sends the ticket T^*, and $\mathcal{P}$ checks it. If passed, $\mathcal{P}$ sends the commitment so that $\mathsf{r}_{\mathcal{I}}$ will be disclosed if $\mathcal{U}$ behaves correctly. Once the user sends the secret value $\mathsf{r}_{\mathcal{U}}$ encrypted through a shared key, then she receives the secret $\mathsf{r}_{\mathcal{I}}$ together with the receipt R^* from $\mathcal{P}$. The service provider $\mathcal{P}$ and the user $\mathcal{U}$ do the following steps:

showTicket $\mathcal{U}$ computes:

1. sends ticket $\mathsf{m}_1 = \mathsf{T}^*$ to $\mathcal{P}$;

verifyTicket $\mathcal{P}$ executes:

1. verifies the ticket signature, $\mathsf{T.Sv}$, $\mathsf{T.Ti}$, and $\mathsf{T.Tv}$;
2. if the verifications fail, $\mathcal{P}$ omits m_1, and aborts the ticket verification;
3. else $\mathcal{P}$ looks for the ticket T^* in the database using $\mathsf{T.Sn}$; and verifies that the ticket has not been spent: $\stackrel{?}{\exists}\, \mathsf{r}_{\mathcal{U}}$ linked to T^* in the DB;
   (a) if $\nexists \mathsf{r}_{\mathcal{U}}$ linked to T^*:
      i. computes $\mathsf{A}_{\mathcal{P}} = PRNG(\mathsf{h}_{\mathsf{K}}) \oplus \mathsf{r}_{\mathcal{I}}$, where $PRNG(\mathsf{h}_{\mathsf{K}})$ is a secure pseudorandom number generator and, $\mathsf{h}_{\mathsf{K}} = hash(\mathsf{K})$ is the seed. Note that K and $\mathsf{r}_{\mathcal{I}}$ are obtained from $\delta_{\mathcal{T},\mathcal{P}}$;
      ii. encrypts $\mathsf{A}_{\mathcal{P}}$ with the public key of the TTP $\mathcal{T}$: $\mathsf{pk}_{\mathcal{T}}(\mathsf{A}_{\mathcal{P}})$;
      iii. assigns $\mathsf{V}_{\mathsf{succ}} = (\mathsf{T.Sn}, \mathsf{flag}_1, \tau_1, \mathsf{pk}_{\mathcal{T}}(\mathsf{A}_{\mathcal{P}}))$, ($\tau_1$ is the verification timestamp). The flag flag_1 indicates that the ticket is valid and has not been spent yet. The signature is noted: ${\mathsf{V}_{\mathsf{succ}}}^* = (\mathsf{V}_{\mathsf{succ}}, \mathsf{sk}_{\mathcal{P}}(hash(\mathsf{V}_{\mathsf{succ}})))$;
      iv. sends $\mathsf{m}_2 = {\mathsf{V}_{\mathsf{succ}}}^*$ to $\mathcal{U}$;
   (b) if $\exists \mathsf{r}_{\mathcal{U}}$ linked to T^*:
      i. assigns $\mathsf{V}_{\mathsf{fail}} = (\mathsf{T.Sn}, \mathsf{r}_{\mathcal{U}}, \mathsf{flag}_0, \tau_1)$. The flag_0 indicates that the ticket has been spent, i.e. is not valid. The signature is noted: ${\mathsf{V}_{\mathsf{fail}}}^* = (\mathsf{V}_{\mathsf{fail}}, \mathsf{sk}_{\mathcal{P}}(hash(\mathsf{V}_{\mathsf{fail}})))$;
      ii. sends $\mathsf{m}_2 = {\mathsf{V}_{\mathsf{fail}}}^*$ to $\mathcal{U}$;

showProof $\mathcal{U}$ executes:

1. verifies $\mathcal{P}$'s signature;
2. if $\mathsf{V_{succ}}^*$ or either $\mathsf{V_{fail}}^*$ are not received, the *Claim* m_2 *Not Received* is called;
   (a) if $\mathsf{V_{fail}}^*$ is received, $\mathcal{U}$ aborts the verification process. If the response is not correct, $\mathcal{U}$ can contact with the TTP to reconsider the situation by calling *Claim* m_2 *Not Received*;
   (b) if $\mathsf{m}_2 = \mathsf{V_{succ}}^*$ is received, $\mathcal{U}$ has to verify the signature and data. If verifications are correct she continues the protocol. Otherwise, $\mathcal{U}$ can contact the TTP by calling *Claim* m_2 *Not Received*;
3. calculates $\mathsf{A}_\mathcal{U} = PRNG(\mathsf{K}) \oplus \mathsf{r}_\mathcal{U}$, using the shared value K as seed;
4. sends $\mathsf{m}_3 = (\mathsf{T.Sn}, \mathsf{A}_\mathcal{U})$ to $\mathcal{P}$;

verifyProof $\mathcal{P}$ follows the next steps:

1. obtains $\mathsf{T.Sn}$, and computes $\mathsf{r}_\mathcal{U} = \mathsf{A}_\mathcal{U} \oplus PRNG(\mathsf{K})$;
2. verifies $\mathsf{h}_{\mathsf{r}_\mathcal{U}} \stackrel{?}{=} hash(\mathsf{r}_\mathcal{U})$;
3. if $\mathsf{r}_\mathcal{U}$ is not received or does not match, the *Claim* m_3 *Not Received* is called;
4. generates τ_2 and verifies it using the ticket expiry date $(\mathsf{T.Ti}, \mathsf{T.Tv})$ and the timestamp τ_1;
5. signs $\mathsf{A}_\mathcal{P}$ approving then the validation with timestamp τ_2: $\mathsf{R} = (\mathsf{A}_\mathcal{P}, \mathsf{T.Sn}, \tau_2)$, and $\mathsf{R}^* = (\mathsf{R}, \mathsf{sk}_\mathcal{P}(hash(\mathsf{R})))$;
6. sends $\mathsf{m}_4 = \mathsf{R}^*$ to $\mathcal{U}$;

getValidationConfirmation $\mathcal{U}$ follows the next steps:

1. checks the signature of R^*;
2. if $\mathsf{r}_\mathcal{I}$ is not received or is incorrect, the *Claim* m_4 *Not Received* is called;
3. computes $\mathsf{r}_\mathcal{I} = \mathsf{A}_\mathcal{P} \oplus PRNG(\mathsf{h_K})$;
4. verifies $\mathsf{h}_{\mathsf{r}_\mathcal{I}} \stackrel{?}{=} hash(\mathsf{r}_\mathcal{I})$;
5. if all verifications are correct, then stores $(\mathsf{R}^*, \mathsf{r}_\mathcal{I})$ together with T^*; else calls *Claim* m_4 *Not Received* to the TTP.

The *Ticket Verification* protocol is a fair exchange protocol with the existence of an offline TTP [11] between the user and the provider of the service (a valid e-ticket is given in exchange for the permission to use the service). This enables dispute resolution protocols in case of incorrect behaviour of the actors so as to preserve the security of the system. In case of dispute, they can contact the TTP following these protocols:

Claim m_2 Not Received. This protocol can be executed if $\mathcal{U}$ sends m_1 and says that she has not received $\mathsf{m}_2 = \mathsf{V_{succ}}^*$ from $\mathcal{P}$.

Claim User $\mathcal{U}$ executes:

1. sends the ticket $\mathsf{m}_1 = \mathsf{T}^*$ to the TTP $\mathcal{T}$;

Response TTP $\mathcal{T}$ follows the next steps:

1. checks the information;
2. if the verification is correct, generates $(\mathsf{T.Sn}, \tau_3)$; then
3. signs the information $\mathsf{m}_5 = ((\mathsf{T.Sn}, \tau_3), \mathsf{sk}_\mathcal{T}(hash((\mathsf{T.Sn}, \tau_3))))$; and
4. sends m_5 to both $\mathcal{U}$ and $\mathcal{P}$. This entails acceptance of $\mathcal{U}$'s sent information and then $\mathcal{P}$ has the responsibility to unblock and send a correct m_2 to continue with the verification phase at sub-phase *verifyTicket*. After that, if the service could not be finally guaranteed, $\mathcal{U}$ could demonstrate to a third party (by showing m_5) that $\mathcal{U}$ behaved correctly and $\mathcal{P}$ was the responsible of the denial of service;

verifyTicketWithTTP Service provider $\mathcal{P}$ executes:

1. executes *verifyTicket* normally;
2. sends m_2 to both $\mathcal{T}$ and $\mathcal{U}$, and continues the *Ticket Verification* steps at point *showProof*;

Claim m_3 Not Received. This protocol can be executed if $\mathcal{P}$ sends m_2 and says that has not received $\mathsf{m}_3 = \mathsf{A}_\mathcal{U}$ (with a correct $\mathsf{r}_\mathcal{U}$ inside) from $\mathcal{U}$.

Claim Provider $\mathcal{P}$ executes:

1. blocks the ticket $\mathsf{T.Sn}$ till the reception of m_3 from $\mathcal{U}$ or m_5 by $\mathcal{T}$;
2. another $\mathsf{T.Sn}'$ received from the same connection could not be accepted and $\mathsf{m}_2 = \mathsf{V_{succ}}^*$ would be repeatedly sent in order to unblock the ticket identified by $\mathsf{T.Sn}$.

Claim m_4 Not Received. This protocol can be executed if $\mathcal{U}$ sends m_3 and says that has not received $\mathsf{m}_4 = \mathsf{R}^*$ (with the contained $\mathsf{r}_\mathcal{I}$) from $\mathcal{P}$.

Claim User $\mathcal{U}$ follows the next steps:

1. sends to the TTP $\mathcal{T}$: $(\mathsf{m}_1, \mathsf{m}_2, \mathsf{m}_3) = (\mathsf{T}^*, \mathsf{V_{succ}}^*, (\mathsf{T.Sn}, \mathsf{A}_\mathcal{U}))$;

Response TTP $\mathcal{T}$ executes:

1. checks the entire information; if verification fails, aborts the claim;
2. computes $\mathsf{A}_\mathcal{P} = PRNG(\mathsf{h_K}) \oplus \mathsf{r}_\mathcal{I}$ using K and $\mathsf{r}_\mathcal{I}$. Note that K and $\mathsf{r}_\mathcal{I}$ can be obtained by decrypting $\delta_{\mathcal{T},\mathcal{P}}$;
3. checks $\mathsf{A}_\mathcal{U}$ obtaining a valid $\mathsf{r}_\mathcal{U}$ that matches with $\mathsf{T.h}_{\mathsf{r}_\mathcal{U}}$;
4. checks $\mathsf{A}_\mathcal{P}$ obtaining a valid $\mathsf{r}_\mathcal{I}$ that matches with $\mathsf{T.h}_{\mathsf{r}_\mathcal{I}}$;
5. if everything is successful, then generates $(\mathsf{T.Sn}, \mathsf{A}_\mathcal{P}, \mathsf{A}_\mathcal{U}, \tau_4)$; otherwise, publishes which entity behaved corruptly;
6. signs the information $\mathsf{m}_6 = ((\mathsf{T.Sn}, \mathsf{A}_\mathcal{P}, \mathsf{A}_\mathcal{U}, \tau_4), \mathsf{sk}_\mathcal{T}(hash((\mathsf{T.Sn}, \mathsf{A}_\mathcal{P}, \mathsf{A}_\mathcal{U}, \tau_4)))$;
7. sends m_6 to $\mathcal{U}$.

3.2 Multiple Providers

The described proposal considers that only one provider is able to give a certain service, enabling then offline verification. Nevertheless, this scenario could be extended to the existence of multiple providers that give a certain service and accept the same ticket in different places, but guaranteeing the control of ticket overspending through online verification between all the providers. In this extended scenario, $\mathsf{sk}_\mathcal{P}$ would be shared between the providers, enabling the access to K and $\mathsf{r}_\mathcal{I}$. There would be also special care to the distribution and control of used tickets (controlled by the existence of $\mathsf{r}_\mathcal{U}$ in the database for that ticket). There should be a central database where all the providers could store all the used tickets, and then the verification would be online by imperative. Expired tickets could be removed from the database for storage efficiency, and, moreover, only ticket serial number could be stored in the database despite storing all the ticket information.

4 Security and Privacy of the System

Proposition 1. *The proposed e-ticketing system preserves authenticity, non-repudiation, integrity and the expiry date of the e-ticket.*

Claim (1). It is computationally unfeasible to make a new fraudulent e-ticket.

Security Argument. A valid e-ticket has the form $\mathsf{T}^* = (\mathsf{T}, \mathsf{Sign}_{\mathcal{I}}(\mathsf{T}))$. Then, the first step that the provider $\mathcal{P}$ does when an e-ticket is received is the verification of the signature. The *Ticket Verification* protocol will continue only if this verification ends correctly; otherwise, $\mathcal{P}$ refuses $\mathcal{U}$'s request. Thus, making a new fraudulent valid e-ticket would be equivalent to break the signature scheme and that would be computationally unfeasible as we have supposed that the issuer $\mathcal{I}$ uses a secure signature scheme.

Claim (2). The issuer can not 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; the signature is a non-repudiation evidence of origin.

Claim (3). The content of the e-ticket can not be modified.

Security Argument. Suppose that someone modifies the content of the ticket, then a new $\mathcal{I}$'s signature has to be generated over the modified content; otherwise, the e-ticket will not be valid. Again, if it is computationally unfeasible to forge the $\mathcal{I}$'s signature, it is unfeasible to modify the content of the e-ticket.

Claim (4). The e-ticket will not be longer valid after the ticket validity time $\mathsf{T.Tv}$.

Security Argument. The provider $\mathcal{P}$ receives the e-ticket from the user at the *Ticket Verification* protocol before allowing access to the service. First of all $\mathcal{P}$ checks the correctness of the e-ticket (obviously that includes the verification of $\mathsf{T.Tv}$). If the verification is not correct $\mathcal{P}$ stops the protocol and the user will have no access to the service. Also, according the *Claim 3*, the user can not tamper $\mathsf{T.Tv}$.

Result 1. *According to the definitions given at Section 3 and the Claims 1, 2, 3 and 4 we can assure that the protocol achieves the properties specified at Proposition 1.*

Proposition 2. *The e-ticketing system described in Section 3 is anonymous. The service offered is revocable anonymous.*

Claim (5). An e-ticket is anonymous.

Security Argument. A valid e-ticket has the following information $\mathsf{T} = (\mathsf{Sn}, \mathsf{Sv}, \mathsf{Pseu}_{\mathcal{U}}, \mathsf{Tv}, \mathsf{Ti}, \mathsf{h}_{\mathsf{r}_{\mathcal{I}}}, \mathsf{h}_{\mathsf{r}_{\mathcal{U}}}, \delta_{\mathcal{T},\mathcal{P}}, \ldots)$. The information related to the user's identity is solely $\mathsf{Pseu}_{\mathcal{U}} = (\mathsf{y}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{I}}(hash(\mathsf{y}_{\mathcal{U}})))$, where $\mathsf{y}_{\mathcal{U}} = \alpha^{\mathsf{x}_{\mathcal{U}}} \pmod{p}$. The user's identity is $\mathsf{x}_{\mathcal{U}}$, thus an enemy has to solve the problem of computing the discrete logarithm to know the identity of the user. Currently no efficient algorithms are known to compute that.

Claim (6). The purchase of an e-ticket is anonymous.

Security Argument. As the protocol in Section 3.1 specifies, the channel between $\mathcal{U}$ and $\mathcal{I}$ of the ticket is anonymous. The protocol uses a Schnorr's ZKP to provide the user identity to the $\mathcal{I}$, so that the issuer can be sure that the connected user who wants to buy the ticket is the right holder of the pseudonym $\mathsf{Pseu}_{\mathcal{U}}$ without disclosing her real identity. Thus, the user does not need to reveal her identity to buy an e-ticket.

Claim (7). A fake user cannot buy an e-ticket impersonating other user.

Security Argument. In order to buy a ticket, the user has to perform a Schnorr's ZKP to prove knowledge of the identity to the issuer without revealing it. The user has to compute w_1 such as $\alpha^{\mathsf{w}_1} \stackrel{?}{=} A_1 \cdot \mathsf{y}_{\mathcal{U}}{}^{\mathsf{c}} \pmod{p}$. As far as any user preserves the privacy of her identity $\mathsf{x}_{\mathcal{U}}$ (which links to $\mathsf{Cert}_{\mathcal{U}}$ through the cooperation of $\mathcal{T}$), anyone else will not be able to compute such w_1. In this case, user can only be accused through $\mathsf{x}_{\mathcal{U}}$ of ticket overspending (supposing that the user keeps $\mathsf{x}_{\mathcal{U}}$ secretly), because she solely has the information to perform the *Ticket Verification* protocol. Thus, the e-ticketing system also preserves the exculpability property.

Result 2. *According to the definitions given at Section 3 and the Claims 5, 6 and 7 we can assert the Proposition 2. The e-ticketing system is anonymous and this anonymity could be revocable in case of a user's fraudulent action. The pseudonym manager $\mathcal{T}$ knows the correspondence between $\mathsf{x}_{\mathcal{U}}$ and $\mathsf{y}_{\mathcal{U}}$ (see Algorithm: 'Pseudonym Renewal'). Therefore, $\mathcal{T}$ could reveal the association between $\mathsf{x}_{\mathcal{U}}$ and $\mathsf{y}_{\mathcal{U}}$ due to law enforcement (e.g. a judge could request the user's identity to $\mathcal{T}$).*

Proposition 3. *The protocol satisfies the property of exculpability.*

Claim (8). The service provider can not falsely accuse the user of ticket overspending.

Security Argument. When the service provider $\mathcal{P}$ receives the message m_1 in step 1 of the *Ticket Verification* protocol *(showTicket)*, $\mathcal{P}$ looks for the ticket that matches with the received serial number in its database. If the ticket had been already spent, the service provider will find the overspending proof $\mathsf{r}_{\mathcal{U}}$ together with the ticket. The service provider has to show this element to accuse the user of overspending. If the user had not validated the ticket before, then the service provider does not have the element ($\mathcal{U}$ will send it in step 3: *showProof*), as the *hash* invertible function is believed to be computationally infeasible, and also there can not exist collisions in this *hash* function, so $\mathcal{P}$ can not falsely accuse the user of overspending. If the service provider, even not being able to prove the overspending, decides to deny the service to the user, the user can contact the TTP in order to solve the situation through *Claim* m_2 *Not Received*.

Claim (9). The user $\mathcal{U}$ is able to prove that she has already validated the ticket.

Security Argument. If a user $\mathcal{U}$ executes successfully the *Ticket Verification* protocol, $\mathcal{U}$ will obtain the exculpability proof $\mathsf{r}_{\mathcal{I}}$. She can use this proof to

demonstrate that the ticket has been validated. If the *Ticket Verification* protocol is stopped and $\mathcal{U}$ does not obtain the exculpability proof after the revelation of $\mathsf{r}_{\mathcal{U}}$, she can execute *Claim* m_4 *Not Received*. This way $\mathcal{U}$ would obtain an alternative exculpability proof from the TTP.

Result 3. *According to the definitions given at Section 3 and the Claims 8 and 9 we can assure that the protocol achieves the property specified at the Proposition 3. The ticket verification process is a fair exchange: any part can obtain the exculpability proof of the other part without revealing its own proof.*

Proposition 4. *The protocol avoids overspending with minimum requirements of persistent connections with a centralized system.*

Claim (10). The protocol avoids overspending.

Security Argument. If a user tries to overspend a ticket, then initiates the verification another time after the first validation of the ticket. The user sends T^* to $\mathcal{P}$ *(showTicket)* and waits for a response. $\mathcal{P}$ executes *(verifyTicket)* and looks for the ticket T^* in its database. Overspending can be prevented; the provider can prove it while $\mathcal{P}$ can not accuse the user of overspending attempts if $\mathcal{P}$ can not provide $\mathsf{r}_{\mathcal{U}}$.

Claim (11). If the ticket can only be validated by one provider then the verification is totally offline.

Security Argument. The provider $\mathcal{P}$ maintains a database with the serial numbers of the e-tickets that have been already validated (together with their exculpability proofs) until their expiry date. With the contents of this database the provider has enough information to decide if $\mathcal{P}$ accepts and validates a new ticket because $\mathcal{P}$ can check both the issuer's signature and the fact that the e-ticket has not been spent before. So the provider does not need to contact to any party during the validation of an e-ticket.

Claim (12). If the ticket can be validated with several providers then the providers must be connected and share a database of spent tickets.

Security Argument. The set of providers maintains a shared database with the serial numbers of the e-tickets that have been already validated (together with their exculpability proofs) until their expiry date. The contents of this database are used by the providers to decide if they accept and validate a new ticket. So the provider does not need connection to the issuer during the verification of an e-ticket, but the set of providers must have a shared database instead.

Result 4. *According to the definitions given at Section 3 and the Claims 10, 11 and 12 we can assure that the protocol achieves Proposition 4. The issuer is offline during the verification phase and the providers contact among them only in some kind of services. In all cases, the protocol prevents ticket overspending.*

5 Conclusions

We have presented an e-ticketing scheme with revocable anonymity, and exculpability as a novel security requirement. Moreover, we have considered the use of personal mobile devices in the system, trying to lighten then the computational requirements on the users' side. A new scenario has been introduced since we assume, for simplicity, that only one provider is able to give a certain service; then, the ticket verification phase is totally offline from the ticket issuer. But, if multiple providers were able to give that service, then ticket verification should be actively centralized (online verification between all the service providers) in order to avoid ticket overspending attempts. An analysis of the security and privacy of the proposal has been also performed, breaking down all the security requirements and evaluating the achievements. The future work is to implement the proposal for mobile devices with contactless communication.

Disclaimer and Acknowledgements

Partial support by the Spanish MICINN(projects TSI2007-65406-C03-01, ARES-CONSOLIDER INGENIO 2010 CSD2007-00004, TSI2007-62986, "RIPUP" TIN2009-11689), the Spanish Ministry of Industry, Commerce and Tourism (project eVerification TSI-020100-2009-720), and the Government of Catalonia (grant 2009 SGR1135) is acknowledged. The authors are solely responsible for the views expressed in this paper, which do not necessarily reflect the position of UNESCO nor commit that organization.

References

1. Bao, F.: A scheme of digital ticket for personal trusted device. In: 15th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC 2004), vol. 4, pp. 3065–3069. IEEE, Los Alamitos (2004)
2. Chaum, D.: Blind signatures for untraceable payments. In: Advances in Cryptology - CRYPTO 1982, pp. 199–203 (1983)
3. Chen, Y.-Y., Chen, C.-L., Jan, J.-K.: A mobile ticket system based on personal trusted device. Wireless Personal Communications: An International Journal 40(4), 569–578 (2007)
4. Dingledine, R., Mathewson, N., Syverson, P.: Tor: The second-generation onion router. In: Proceedings of the 13th USENIX Security Symposium (2004)
5. Elliot, J.: The one-card trick multi-application smart card e-commerce prototypes. Computing & Control Engineering Journal 10(3), 121–128 (1999); IET
6. Fujimura, K., Kuno, H., Terada, M., Matsuyama, K., Mizuno, Y., Sekine, J.: Digital-ticket-controlled digital ticket circulation. In: 8th USENIX Security Symposium, pp. 229–240. USENIX (1999)
7. Haneberg, D.: Electronic ticketing: risks in e-commerce applications. In: Digital Excellence, pp. 55–66. Springer, Heidelberg (2008) ISBN 3540726209
8. Haneberg, D., Stenzel, K., Reif, W.: Electronic-onboard-ticketing: Software challenges of an state-of-the-art m-commerce application. In: Pousttchi, K., Turowski, K. (eds.) Workshop Mobile Commerce. Lecture Notes in Informatics (LNI), vol. 42, pp. 103–113. Gesellschaft für Informatik, GI (2004)

9. Heydt-Benjamin, T.S., Chae, H.-J., Defend, B., Fu, K.: Privacy for public transportation. In: Danezis, G., Golle, P. (eds.) PET 2006. LNCS, vol. 4258, pp. 1–19. Springer, Heidelberg (2006)
10. Gans, G.K., Hoepman, J.-H., Garcia, F.D.: A practical attack on the mifare classic. In: Grimaud, G., Standaert, F.-X. (eds.) CARDIS 2008. LNCS, vol. 5189, pp. 267–282. Springer, Heidelberg (2008)
11. Kremer, S., Markowitch, O., Zhou, J.: An intensive survey of fair non-repudiation protocols. Computer Communications 25, 1606–1621 (2002)
12. Kuramitsu, K., Sakamura, K.: Electronic tickets on contactless smartcard database. In: Hameurlain, A., Cicchetti, R., Traunmüller, R. (eds.) DEXA 2002. LNCS, vol. 2453, pp. 392–402. Springer, Heidelberg (2002)
13. Kuramitsu, K., Murakami, T., Matsuda, H., Sakamura, K.: Ttp: Secure acid transfer protocol for electronic ticket between personal tamper-proof devices. In: 24th Annual International Computer Software and Applications Conference (COMPSAC 2000), Taipei, Taiwan, vol. 24, pp. 87–92 (October 2000)
14. Matsuo, S., Ogata, W.: Electronic ticket scheme for its. IEICE Transactions on Fundamentals of Electronics Communications and Computer Sciences E86A(1), 142–150 (2003)
15. Patel, B., Crowcroft, J.: Ticket based service access for the mobile user. In: Proceedings of the 3rd Annual ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM 1997), Budapest, Hungary, pp. 223–233 (1997)
16. Quercia, D., Hailes, S.: Motet: Mobile transactions using electronic tickets. In: Proceedings 1st International Conference on Security and Privacy for Emerging Areas in Communications Networks, Athens, Greece, vol. 24, pp. 374–383 (September 2005)
17. Schnorr, C.-P.: Efficient signature generation by smart cards. J. Cryptology 4(3), 161–174 (1991)
18. Siu, W.I., Guo, Z.S.: Application of electronic ticket to online trading with smart card technology. In: Proceedings of the 6th INFORMS Conference on Information Systems and Technology (CIST 2001), Florida (US), pp. 222–239 (2001)
19. Siu, W.I., Guo, Z.S.: The secure communication protocol for electronic ticket management system. In: 8th Asia-Pacific Software Engineering Conference (APSEC 2001), University of Macau (2001)